KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation

Michele Borassi, Emanuele Natale

KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation*

Michele Borassi[1] and Emanuele Natale[2]

[1]IMT Insitute for Advanced Studies, 55100 Lucca, Italy, michele.borassi@imtlucca.it
[2]Sapienza University of Rome, 00185 Roma, Italy, natale@di.uniroma1.it

15th August 2016

Abstract

We present KADABRA, a new algorithm to approximate betweenness centrality in directed and undirected graphs, which significantly outperforms all previous approaches on real-world complex networks. The efficiency of the new algorithm relies on two new theoretical contributions, of independent interest.

The first contribution focuses on sampling shortest paths, a subroutine used by most algorithms that approximate betweenness centrality. We show that, on realistic random graph models, we can perform this task in time $|E|^{\frac{1}{2}+o(1)}$ with high probability, obtaining a significant speedup with respect to the $\Theta(|E|)$ worst-case performance. We experimentally show that this new technique achieves similar speedups on real-world complex networks, as well.

The second contribution is a new rigorous application of the adaptive sampling technique. This approach decreases the total number of shortest paths that need to be sampled to compute all betweenness centralities with a given absolute error, and it also handles more general problems, such as computing the k most central nodes. Furthermore, our analysis is general, and it might be extended to other settings, as well.

1 Introduction

In this work we focus on estimating the *betweenness centrality*, which is one of the most famous measures of *centrality* for nodes and edges of real-world complex networks [21, 33]. The rigorous definition of betweenness centrality has its roots in sociology, dating back to the Seventies, when Freeman formalized the informal concept discussed in the previous decades in different scientific communities [6, 43, 42, 19, 15], although the definition already appeared in [4]. Since then, this notion has been very successful in network science [48, 34, 25, 33].

A probabilistic way to define the betweenness centrality[1] $\mathrm{bc}(v)$ of a node v in a graph $G = (V, E)$ is the following. We choose two nodes s and t, and we go from s to t through a shortest path π; if the choices of s, t and π are made uniformly at random, the betweenness centrality of a node v is the probability that we pass through v.

In a seminal paper [16], Brandes showed that it is possible to exactly compute the betweenness centrality of all the nodes in a graph in time $\mathcal{O}(mn)$, where n is the number of nodes and m is the number of edges. A corresponding lower bound was proved in [13]: if we

*This work was done while the authors were visiting the Simons Institute for the Theory of Computing.
[1]As explained in see Section 2, to simplify notation we consider the *normalized* betweenness centrality.

are able to compute the betweenness centrality of a single node in time $\mathcal{O}(mn^{1-\epsilon})$ for some $\epsilon > 0$, then the Strong Exponential Time Hypothesis [26] is false.

This result further motivates the rich line of research on computing approximations of betweenness centrality, with the goal of trading precision with efficiency. The main idea is to define a probability distribution over the set of all paths, by choosing two uniformly random nodes s, t, and then a uniformly distributed st-path $\boldsymbol{\pi}$, so that $\Pr(v \in \boldsymbol{\pi}) = \mathrm{bc}(v)$. As a consequence, we can approximate $\mathrm{bc}(v)$ by sampling paths $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\tau$ according to this distribution, and estimating $\tilde{\boldsymbol{b}}(v) := \frac{1}{\tau}\sum_{i=1}^{\tau} \boldsymbol{X}_i(v)$, where $\boldsymbol{X}_i(v) = 1$ if $v \in \boldsymbol{\pi}_i$ (and $v \neq s, t$), 0 otherwise.

The tricky part of this approach is to provide probabilistic guarantees on the quality of this approximation: the goal is to obtain a $1-\delta$ confidence interval $\boldsymbol{I}(v) = [\tilde{\boldsymbol{b}}(v) - \lambda_L, \tilde{\boldsymbol{b}}(v) + \lambda_U]$ for $\mathrm{bc}(v)$, which means that $\Pr(\forall v \in V, \mathrm{bc}(v) \in \boldsymbol{I}(v)) \geq 1 - \delta$. Thus, the research for approximating betweenness centrality has been focusing on obtaining, as fast as possible, the smallest possible $\boldsymbol{I}$.

Our Contribution

In this work, we propose a new and faster algorithm to approximate betweenness centrality in directed and undirected graphs, named KADABRA. In the standard task of approximating betweenness centralities with absolute error at most λ, we show that, on average, the new algorithm is more than 100 times faster than the previous ones, on graphs with approximately 10 000 nodes. Moreover, differently from previous approaches, our algorithm can perform more general tasks, since it does not need all confidence intervals to be equal. As an example, we consider the computation of the k most central nodes: all previous approaches compute all centralities with an error λ, and use this approximation to obtain the ranking. Conversely, our approach allows us to use small confidence interval only when they are needed, and allows bigger confidence intervals for nodes whose centrality values are “well separated”. This way, we can compute for the first time an approximation of the k most central nodes in networks with millions of nodes and hundreds of millions of edges, like the Wikipedia citation network and the IMDB actor collaboration network.

Our results rely on two main theoretical contributions, which are interesting in their own right, since their generality naturally extends to other applications.

Balanced bidirectional breadth-first search. By leveraging on recent advanced results, we prove that, on many realistic random models of real-world complex networks, it is possible to sample a random path between two nodes s and t in time $m^{\frac{1}{2}+o(1)}$ if the degree distribution has finite second moment, or $m^{\frac{4-\beta}{2}+o(1)}$ if the degree distribution is power law with exponent $2 < \beta < 3$. The models considered are the Configuration Model [11], and all Rank-1 Inhomogeneous Random Graph models [45, Chapter 3], such as the Chung-Lu model [32], the Norros-Reittu model [35], and the Generalized Random Graph [45, Chapter 3]. Our proof techniques have the merit of adopting a unified approach that simultaneously works in all models considered. These models well represent metric properties of real-world networks [14]: indeed, our results are confirmed by practical experiments.

The algorithm used is simply a balanced bidirectional BFS (bb-BFS): we perform a BFS from each of the two endpoints s and t, in such a way that the two BFSs are likely to explore about the same number of edges, and we stop as soon as the two BFSs “touch each other”. Rather surprisingly, this technique was never implemented to approximate betweenness centrality, and it is rarely used in the experimental algorithm community. Our theoretical analysis provides a clear explanation of the reason why this technique improves over the standard BFS: this means that many state-of-the-art algorithm for real-world complex networks can be improved by the bb-BFS.

Adaptive sampling made rigorous. To speed up the estimation of the betweenness centrality, previous work make use of the technique of adaptive sampling, which consists in testing during the execution of the algorithm whether some condition on the sample obtained so far has been met, and terminating the execution of the algorithm as soon as this happens. However, this technique introduces a subtle stochastic dependence between the time in which the algorithm terminates and the correctness of the given output, which previous papers claiming a formal analysis of the technique did not realize (see Section 3 for details). With an argument based on martingale theory, we provide a general analysis of such useful technique. Through this result, we do not only improve previous estimators, but we also make it possible to define more general stopping conditions, that can be decided "on the fly": this way, with little modifications, we can adapt our algorithm to perform more general tasks than previous ones.

To better illustrate the power of our techniques, we focus on the unweighted, static graphs, and to the centrality of nodes. However, our algorithm can be easily adapted to compute the centrality of edges, to handle weighted graphs and, since its core part consists merely in sampling paths, we conjecture that it may be coupled with the existing techniques in [9] to handle dynamic graphs.

Related Work

Computing Betweenness Centrality. With the recent event of big data, the major shortcoming of betweenness centrality has been the lack of efficient methods to compute it [16]. In the worst case, the best exact algorithm to compute the centrality of all the nodes is due to Brandes [16], and its time complexity is $\mathcal{O}(mn)$: the basic idea of the algorithm is to define the dependency $\delta_s(v) = \sum_{t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, which can be computed in time $\mathcal{O}(m)$, for each $v \in V$ (we denote by $\sigma_{st}(v)$ the number of shortest paths from s to t passing through v, and by σ_{st} the number of st-shortest paths). In [13], it is also shown that Brandes algorithm is almost optimal on sparse graphs: an algorithm that computes the betweenness centrality of a single vertex in time $\mathcal{O}(mn^{1-\epsilon})$ falsifies widely believed complexity assumptions, such as the Strong Exponential Time Hypothesis [26], the Orthogonal Vector conjecture [2], or the Hitting Set conjecture [49]. Corresponding results in the dense, weighted case are available in [1]: computing the betweenness centrality exactly is as hard as computing the All Pairs Shortest Path, and computing an approximation with a given relative error is as hard as computing the diameter. For both these problems, there is no algorithm with running-time $\mathcal{O}(n^{3-\epsilon})$, for any $\epsilon > 0$. This shows that, for dense graphs, having an additive approximation rather than a multiplicative one is essential for a provably fast algorithm to exist. These negative results further motivate the already rich line of research on approaches that overcome this barrier. A first possibility is to use heuristics, that do not provide analytical guarantees on their performance [41, 23, 46]. Another line of research has defined variants of betweenness centrality, that might be easier to compute [17, 36, 20]. Finally, a third line of research has investigated approximation algorithms, which trade accuracy for speed [27, 18, 25, 29]. Our work follows the latter approach. The first approximation algorithm proposed in the literature [27] adapts Eppstein and Wang's approach for computing closeness centrality [22], using Hoeffding's inequality and the union bound technique. This way, it is possible to obtain an estimate of the betweenness centrality of every node that is correct up to an additive error λ with probability δ, by sampling $\mathcal{O}(\frac{D^2}{\lambda^2} \log \frac{n}{\delta})$ nodes, where D is the diameter of the graph. In [25], it is shown that this can lead to an overestimation. Riondato and Kornaropoulos improve this sampling-based approach by sampling single shortest paths instead of the whole dependency of a node [39], introducing the use of the VC-dimension. As a result, the number of samples is decreased to $\frac{c}{\lambda^2}(\lfloor \log_2(\mathrm{VD}-2) \rfloor + 1 + \log(\frac{1}{\delta}))$, where VD is the vertex diameter, that is, the minimum number of nodes in a shortest path in G (it can be different from $D+1$ if the graph is weighted). This use of the VC-dimension is further developed and generalized in [40]. Finally, many of these results were adapted to handle dynamic networks [9, 40].

Approximating the top-k betweenness centrality set. Let us order the nodes $v_1, ..., v_n$ such that $\mathrm{bc}(v_1) \geq ... \geq \mathrm{bc}(v_n)$ and define $TOP(k) = \{(v_i, \mathrm{bc}(v_i)) : i \leq k\}$. In [39] and [40], the authors provide an algorithm that, for any given δ, ϵ, with probability $1-\delta$ outputs a set $\widetilde{TOP}(k) = \{(v_i, \tilde{\boldsymbol{b}}(v_i))\}$ such that: i) If $v \in TOP(k)$ then $v \in \widetilde{TOP}(k)$ and $|\mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v)| \leq \epsilon\,\mathrm{bc}(v)$; ii) If $v \in \widetilde{TOP}(k)$ but $v \notin TOP(k)$ then $\tilde{\boldsymbol{b}}(v) \leq (\mathbf{b}_k - \epsilon)(1+\epsilon)$ where $\mathbf{b}_k$ is the k-th largest betweenness given by a preliminary phase of the algorithm.

Adaptive sampling. In [5, 40], the number of samples required is substantially reduced using the adaptive sampling technique introduced by Lipton and Naughton in [31, 30]. Let us clarify that, by adaptive sampling, we mean that the termination of the sampling process depends on the sample observed so far (in other cases, the same expression refers to the fact that the distribution of the new samples is a function of the previous ones [3], while the sample size is fixed in advance). Except for [37], previous approaches tacitly assume that there is little dependency between the stopping time and the correctness of the output: indeed, they prove that, for each *fixed* τ, the probability that the estimate is wrong at time τ is below δ. However, the stopping time $\boldsymbol{\tau}$ is a random variable, and in principle there might be dependency between the event $\boldsymbol{\tau} = \tau$ and the event that the estimate is correct at time τ. As for [37], they consider a specific stopping condition and their proof technique does not seem to extend to other settings. For a more thorough discussion of this issue, we defer the reader to Section 3.

Bidirectional BFS. The possibility of speeding up a breadth-first search for the shortest-path problem by performing, at the same time, a BFS from the final end-point, has been considered since the Seventies [38]. Unfortunately, because of the lack of theoretical results dealing with its efficiency, the bidirectional BFS has apparently not been considered a fundamental heuristic improvement [28]. However, in [39] (and in some public talks by M. Riondato), the bidirectional BFS was proposed as a possible way to improve the performance of betweenness centrality approximation algorithms.

Structure of the Paper

In Section 2, we describe our algorithm, and in Section 3 we discuss the main difficulty of the adaptive sampling, and the reasons why our techniques are not affected. In Section 4, we define the balanced bidirectional BFS, and we sketch the proof of its efficiency on random graphs. In Section 5, we show that our algorithm can be adapted to compute the k most central nodes. In Section 6 we experimentally show the effectiveness of our new algorithm. Finally, all our proofs are in the appendix.

2 Algorithm Overview

To simplify notation, we always consider the *normalized* betweenness centrality of a node v, which is defined by:

$$\mathrm{bc}(v) = \frac{1}{n(n-1)} \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest paths between s and t, and $\sigma_{st}(v)$ is the number of shortest paths between s and t that pass through v. Furthermore, to simplify the exposition, we use bold symbols to denote random variables, and light symbols to denote deterministic quantities. On the same line of previous works, our algorithm samples random paths $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\tau$, where $\boldsymbol{\pi}_i$ is chosen by selecting uniformly at random two nodes s, t, and then selecting uniformly at random one of the shortest paths from s to t. Then, it estimates $\mathrm{bc}(v)$ with $\tilde{\boldsymbol{b}}(v) := \frac{1}{\tau}\sum_{i=1}^{\tau} \boldsymbol{X}_i(v)$, where $\boldsymbol{X}_i(v) = 1$ if $v \in \boldsymbol{\pi}_i$, 0 otherwise. By definition of $\boldsymbol{\pi}_i$, $\mathbb{E}\left[\tilde{\boldsymbol{b}}(v)\right] = \mathrm{bc}(v)$.

The tricky part is to bound the distance between $\tilde{\boldsymbol{b}}(v)$ and its expected value. With a straightforward application of Hoeffding's inequality (Lemma 5 in the appendix), it is possible to prove that $\Pr\left(\left|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)\right| \geq \lambda\right) \leq 2e^{-2\tau\lambda^2}$. A direct application of this inequality considers a union bound on all possible nodes v, obtaining $\Pr(\exists v \in V, |\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| \geq \lambda) \leq 2ne^{-2\tau\lambda^2}$. This means that the algorithm can safely stop as soon as $2ne^{-2\tau\lambda^2} \leq \delta$, that is, after $\tau = \frac{1}{2\lambda^2}\log(\frac{2n}{\delta})$ steps.

In order to improve this idea, we can start from Lemma 7 in the appendix, instead of Hoeffding inequality, obtaining that $\Pr\left(\left|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)\right| \geq \lambda\right) \leq 2\exp(-\frac{\tau\lambda^2}{2(\mathrm{bc}(v)+\lambda/3)})$.

If we assume the error λ to be small, this inequality is stronger than the previous one for all values of $\mathrm{bc}(v) < \frac{1}{4}$ (a condition which holds for almost all nodes, in almost all graphs considered). However, in order to apply this inequality, we have to deal with the fact that we do not know $\mathrm{bc}(v)$ in advance, and hence we do not know when to stop. Intuitively, to solve this problem, we make a "change of variable", and we rewrite the previous inequality as

$$\Pr\left(\mathrm{bc}(v) \leq \tilde{\boldsymbol{b}}(v) - f\right) \leq \delta_L^{(v)} \quad \text{and} \quad \Pr\left(\mathrm{bc}(v) \geq \tilde{\boldsymbol{b}}(v) + g\right) \leq \delta_U^{(v)}, \tag{1}$$

for some functions $f = f(\tilde{\boldsymbol{b}}(v), \delta_L^{(v)}, \tau), g = g(\tilde{\boldsymbol{b}}(v), \delta_U^{(v)}, \tau)$. Our algorithm fixes at the beginning the values $\delta_L^{(v)}, \delta_U^{(v)}$ for each node v, and, at each step, it tests if $f(\tilde{\boldsymbol{b}}(v), \delta_L^{(v)}, \tau)$ and $g(\tilde{\boldsymbol{b}}(v), \delta_U^{(v)}, \tau)$ are small enough. If this condition is satisfied, the algorithm stops. Note that this approach lets us define very general stopping conditions, that might depend on the centralities computed until now, on the single nodes, and so on.

Remark 1. *Instead of fixing the values $\delta_L^{(v)}, \delta_U^{(v)}$ at the beginning, one might want to decide them during the algorithm, depending on the outcome. However, this is not formally correct, because of dependency issues (for example,* (1) *does not even make sense, if $\delta_L^{(v)}, \delta_U^{(v)}$ are random). Finding a way to overcome this issue is left as a challenging open problem (more details are provided in Section 3).*

In order to implement this idea, we still need to solve an issue: (1) holds for each *fixed* time τ, but the stopping time of our algorithm is a random variable $\boldsymbol{\tau}$, and there might be dependency between the value of $\boldsymbol{\tau}$ and the probability in (1). To this purpose, we use a stronger inequality (Theorem 8 in the appendix), that holds even if $\boldsymbol{\tau}$ is a random variable. However, to use this inequality, we need to assume that $\boldsymbol{\tau} < \omega$ for some deterministic ω: in our algorithm, we choose $\omega = \frac{c}{\lambda^2}\left(\lfloor \log_2(\mathrm{VD}-2)\rfloor + 1 + \log\left(\frac{2}{\delta}\right)\right)$, because, by the results in [39], after ω samples, the maximum error is at most λ, with probability $1 - \frac{\delta}{2}$. Furthermore, also f and g should be modified, since they now depend on the value of ω. The pseudocode of the algorithm obtained is available in Algorithm 1 (as was done in previous approaches, we can easily parallelize the while loop in Line 5).

The correctness of the algorithm follows from the following theorem, which is the base of our adaptive sampling, and which we prove in Appendix C (where we also define the functions f and g).

Theorem 2. *Let $\tilde{\boldsymbol{b}}(v)$ be the output of Algorithm 1, and let $\boldsymbol{\tau}$ be the number of samples at the end of the algorithm. Then, with probability $1 - \delta$, the following conditions hold:*

- *if $\boldsymbol{\tau} = \omega$, $|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| < \lambda$ for all v;*
- *if $\boldsymbol{\tau} < \omega$, $-f(\boldsymbol{\tau}, \tilde{\boldsymbol{b}}(v), \delta_L^{(v)}, \omega) \leq \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \leq g(\boldsymbol{\tau}, \tilde{\boldsymbol{b}}(v), \delta_U^{(v)}, \omega)$ for all v.*

Algorithm 1: our algorithm for approximating betweenness centrality.

Input : a graph $G = (V, E)$

Output : for each $v \in V$, an approximation $\tilde{\boldsymbol{b}}(v)$ of $\mathrm{bc}(v)$ such that $\Pr\left(\forall v, |\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| \le \lambda\right) \ge 1 - \delta$

$\omega \leftarrow \frac{c}{\lambda^2}\left(\lfloor \log_2(\mathrm{VD}-2) \rfloor + 1 + \log\left(\frac{2}{\delta}\right)\right)$;

$(\delta_L^{(v)}, \delta_U^{(v)}) \leftarrow$ `computeDelta`();

$\tau \leftarrow 0$;

foreach $v \in V$ **do** $\tilde{\boldsymbol{b}}(v) \leftarrow 0$

while $\tau < \omega$ *and not* `haveToStop` $(\tilde{\boldsymbol{b}}, \delta_L, \delta_U, \omega, \tau)$ **do**

  $\boldsymbol{\pi} =$ `samplePath`();

  foreach $v \in \boldsymbol{\pi}$ **do** $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v) + 1$ $\tau \leftarrow \tau + 1$;

end

foreach $v \in V$ **do** $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v)/\tau$ **return** $\tilde{\boldsymbol{b}}$

Remark 3. *This theorem says that, at the beginning of the algorithm, we know that, with probability $1-\delta$, one of the two conditions will hold when the algorithm stops, independently of the final value of $\boldsymbol{\tau}$. This is essential to avoid the stochastic dependence that we discuss in Section 3.*

In order to apply this theorem, we choose λ such that our goal is reached if all centralities are known with error at most λ. Then, we choose the function `haveToStop` in a way that our goal is reached if the stopping condition is satisfied. This way, our algorithm is correct, both if $\boldsymbol{\tau} = \omega$ and if $\boldsymbol{\tau} < \omega$. For example, if we want to compute all centralities with bounded absolute error, we simply choose λ as the bound we want to achieve, and we plug the stopping condition $f, g \le \lambda$ in the function `haveToStop`. Instead, if we want to compute an approximation of the k most central nodes, we need a different definition of f and g, which is provided in Section 5.

To complete the description of this algorithm, we need to specify the following functions.

`computeDelta` The algorithm works for any choice of the $\delta_L^{(v)}, \delta_U^{(v)}$s, but a good choice yields better running times. We propose a heuristic way to choose them in Appendix D.

`samplePath` In order to sample a path between two random nodes s and t, we use a balanced bidirectional BFS, which is defined in Appendix E.

3 Adaptive Sampling

In this section, we highlight the main technical difficulty in the formalization of adaptive sampling, which previous works claiming analogous results did not address. Furthermore, we sketch the way we overcome this difficulty: our argument is quite general, and it could be easily adapted to formalize these claims.

As already said, the problem is the stochastic dependence between the time $\boldsymbol{\tau}$ in which the algorithm terminates and the event $\boldsymbol{A}_\tau =$ "at time τ, the estimate is within the required distance from the true value", since both $\boldsymbol{\tau}$ and $\boldsymbol{A}_\tau$ are functions of the same random sample. Since it is typically possible to prove that $\Pr(\neg \boldsymbol{A}_\tau) \le \delta$ for every fixed τ, one may be tempted to argue that also $\Pr(\neg \boldsymbol{A}_{\boldsymbol{\tau}}) \le \delta$, by applying these inequalities at time $\boldsymbol{\tau}$. However, this is not correct: indeed, if we have no assumptions on $\boldsymbol{\tau}$, $\boldsymbol{\tau}$ could even be defined as the smallest τ such that $\boldsymbol{A}_\tau$ does not hold!

More formally, if we want to link $\Pr(\neg \boldsymbol{A}_{\boldsymbol{\tau}})$ to $\Pr(\neg \boldsymbol{A}_\tau)$, we have to use the law of total probability, that says that:

$$\Pr(\neg \boldsymbol{A}_{\boldsymbol{\tau}}) = \sum_{\tau=1}^{\infty} \Pr(\neg \boldsymbol{A}_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} = \tau) \Pr(\boldsymbol{\tau} = \tau) \tag{2}$$

$$= \Pr(\neg \boldsymbol{A}_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} < \tau) \Pr(\boldsymbol{\tau} < \tau) + \Pr(\neg \boldsymbol{A}_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} \geq \tau) \Pr(\boldsymbol{\tau} \geq \tau). \tag{3}$$

Then, if we want to bound $\Pr(\neg \boldsymbol{A}_{\boldsymbol{\tau}})$, we need to assume that

$$\Pr(\neg A_{\boldsymbol{\tau}} \mid \boldsymbol{\tau} = \tau) \leq \Pr(\neg A_{\tau}) \quad \text{or that} \quad \Pr(\neg A_{\tau} \mid \boldsymbol{\tau} \geq \tau) \leq \Pr(\neg A_{\tau}), \tag{4}$$

which would allow to bound (2) or (3) from above. The equations in (4) are implicitly assumed to be true in previous works adopting adaptive sampling techniques. Unfortunately, because of the stochastic dependence, it is quite difficult to prove such inequalities, even if some approaches managed to overcome these difficulties [37].

For this reason, our proofs avoid dealing with such relations: in the proof of Theorem 2, we fix a deterministic time ω, we impose that $\boldsymbol{\tau} \leq \omega$, and we apply the inequalities with $\tau = \omega$. Then, using martingale theory, we convert results that hold at time ω to results that hold at the stopping time $\boldsymbol{\tau}$ (see Appendix C).

4 Balanced Bidirectional BFS

A major improvement of our algorithm, with respect to previous counterparts, is that we sample shortest paths through a balanced bidirectional BFS, instead of a standard BFS. In this section, we describe this technique, and we bound its running time on realistic models of random graphs, with high probability. The idea behind this technique is very simple: if we need to sample a uniformly random shortest path from s to t, instead of performing a full BFS from s until we reach t, we perform at the same time a BFS from s and a BFS from t, until the two BFSs touch each other (if the graph is directed, we perform a "forward" BFS from s and a "backward" BFS from t).

More formally, assume that we have visited up to level l_s from s and to level l_t from t, let $\Gamma^{l_s}(s)$ be the set of nodes at distance l_s from s, and similarly let $\Gamma^{l_t}(t)$ be the set of nodes at distance l_t from t. If $\sum_{v \in \Gamma^{l_s}(s)} \deg(v) \leq \sum_{v \in \Gamma^{l_t}(t)} \deg(v)$, we process all nodes in $\Gamma^{l_s}(s)$, otherwise we process all nodes in $\Gamma^{l_t}(t)$ (since the time needed to process level l_s is proportional to $\sum_{v \in \Gamma^{l_s}(s)} \deg(v)$, this choice minimizes the time needed to visit the next level). Assume that we are processing the node $v \in \Gamma^{l_s}(s)$ (the other case is analogous). For each neighbor w of v we do the following:

- if w was never visited, we add w to $\Gamma^{l_s+1}(s)$;
- if w was already visited in the BFS from s, we do not do anything;
- if w was visited in the BFS from t, we add the edge (v, w) to the set Π of candidate edges in the shortest path.

After we have processed a level, we stop if $\Gamma^{l_s}(s)$ or $\Gamma^{l_t}(t)$ is empty (in this case, s and t are not connected), or if Π is not empty. In the latter case, we select an edge from Π, so that the probability of choosing the edge (v, w) is proportional to $\sigma_{sv}\sigma_{wt}$ (we recall that σ_{xy} is the number of shortest paths from x to y, and it can be computed during the BFS as in [18]). Then, the path is selected by considering the concatenation of a random path from s to v, the edge (v, w), and a random path from w to t. These random paths can be easily chosen by backtracking, as shown in [39] (since the number of paths might be exponential in the input size, in order to avoid pathological cases, we assume that we can perform arithmetic operations in $\mathcal{O}(1)$ time).

4.1 Analysis on Random Graph

In order to show the effectiveness of the balanced bidirectional BFS, we bound its running time in several models of random graphs: the Configuration Model (CM, [11]), and Rank-1 Inhomogeneous Random Graph models (IRG, [45, Chapter 3]), such as the Chung-Lu model [32], the Norros-Reittu model [35], and the Generalized Random Graph [45, Chapter 3]. In these models, we fix the number n of nodes, and we give a weight ρ_u to each node. In the CM, we create edges by giving ρ_u half-edges to each node u, and pairing these half-edges uniformly at random; in IRG we connect each pair of nodes (u, v) independently with probability close to $\rho_u\rho_v/\sum_{w\in V}\rho_w$. With some technical assumptions discussed in Appendix E, we prove the following theorem.

Theorem 4. *Let G be a graph generated through the aforementioned models. Then, for each fixed $\epsilon > 0$, and for each pair of nodes s, t, w.h.p., the time needed to compute an st-shortest path through a bidirectional BFS is $\mathcal{O}(n^{\frac{1}{2}+\epsilon})$ if the degree distribution λ has finite second moment, $\mathcal{O}(n^{\frac{4-\beta}{2}+\epsilon})$ if λ is a power law distribution with $2 < \beta < 3$.*

Sketch of proof. The idea of the proof is that the time needed by a bidirectional BFS is proportional to the number of visited edges, which is close to the sum of the degrees of the visited nodes, which are very close to their weights. Hence, we have to analyze the weights of the visited edges: for this reason, if V' is a subset of V, we define the volume of V' as $\rho_{V'} = \sum_{v\in V'}\rho_v$.

Our visit proceeds by "levels" in the BFS trees from s and t: if we never process a level with total weight at least $n^{\frac{1}{2}+\epsilon}$, since the diameter is $\mathcal{O}(\log n)$, the volume of the set of processed vertices is $\mathcal{O}(n^{\frac{1}{2}+\epsilon}\log n)$, and the number of visited edges cannot be much bigger (for example, this happens if s and t are not connected). Otherwise, assume that, at some point, we process a level l_s in the BFS from s with total weight $n^{\frac{1}{2}+\epsilon}$: then, the corresponding level l_t in the BFS from t has also weight $n^{\frac{1}{2}+\epsilon}$ (otherwise, we would have expanded from t, because weights and degrees are strongly correlated). We use the "birthday paradox": levels $l_s + 1$ in the BFS from s, and level $l_t + 1$ in the BFS from t are random sets of nodes with size close to $n^{\frac{1}{2}+\epsilon}$, and hence there is a node that is common to both, w.h.p.. This means that the time needed by the bidirectional BFS is proportional to the volume of all levels in the BFS tree from s, until l_s, plus the volume of all levels in the BFS tree from t, until l_t (note that we do not expand levels $l_s + 1$ and $l_t + 1$). All levels except the last have volume at most $n^{\frac{1}{2}+\epsilon}$, and there are $\mathcal{O}(\log n)$ such levels because the diameter is $\mathcal{O}(\log n)$: it only remains to estimate the volume of the last level.

By definition of the models, the probability that a node v with weight ρ_v belongs to the last level is about $\frac{\rho_v\rho_{\mathbf{\Gamma}^{l_s-1}(s)}}{M} \le \rho_v n^{-\frac{1}{2}+\epsilon}$: hence, the expected volume of $\mathbf{\Gamma}^{l_s}(s)$ is at most $\sum_{v\in V}\rho_v \Pr(v \in \mathbf{\Gamma}^{l_s-1}(s)) \le \sum_{v\in V}\rho_v^2 n^{-\frac{1}{2}+\epsilon}$. Through standard concentration inequalities, we prove that this random variable is concentrated: hence, we only need to compute this expected value. If the degree distribution has finite second moment, then $\sum_{v\in V}\rho_v^2 = \mathcal{O}(n)$, concluding the proof. If the degree distribution is power law with $2 < \beta < 3$, then we have to consider separately nodes v such that $\rho_v < n^{\frac{1}{2}}$ and such that $\rho_v > n^{\frac{1}{2}}$. In the first case, $\sum_{\rho_v<n^{\frac{1}{2}}}\rho_v^2 \approx \sum_{d=0}^{n^{\frac{1}{2}}} nd^2\lambda(d) \approx \sum_{d=0}^{n^{\frac{1}{2}}} nd^{2-\beta} \approx n^{1+\frac{3-\beta}{2}}$. In the second case, we prove that the volume of the set of nodes with weight bigger than $n^{\frac{1}{2}}$ is at most $n^{\frac{4-\beta}{2}}$. Hence, the total volume of $\mathbf{\Gamma}^{l_s}(s)$ is at most $n^{-\frac{1}{2}+\epsilon}n^{1+\frac{3-\beta}{2}} + n^{\frac{4-\beta}{2}} \approx n^{\frac{4-\beta}{2}}$.

□

5 Computing the k Most Central Nodes

Differently from previous works, our algorithm is more flexible, making it possible to compute the betweenness centrality of different nodes with different precision. This feature can be exploited if we only want to rank the nodes: for instance, if v is much more central than all the other nodes, we do not need a very precise estimation on the centrality of v to say that it is the top node. Following this idea, in this section we adapt our approach to the approximation of the ranking of the k most central nodes: as far as we know, this is the first approach which computes the ranking without computing a λ-approximation of all betweenness centralities, allowing significant speedups. Clearly, we cannot expect our ranking to be always correct, otherwise the algorithm does not terminate if two of the k most central nodes have the same centrality. For this reason, the user fixes a parameter λ, and, for each node v, the algorithm does one of the following:

- it provides the exact position of v in the ranking;
- it guarantees that v is not in the top-k;
- it provides a value $\tilde{\boldsymbol{b}}(v)$ such that $|\operatorname{bc}(v) - \tilde{\boldsymbol{b}}(v)| \leq \lambda$.

In other words, similarly to what is done in [39], the algorithm provides a set of $k' \geq k$ nodes containing the top-k nodes, and for each pair of nodes v, w in this subset, either we can rank correctly v and w, or v and w are almost even, that is, $|\operatorname{bc}(v) - \operatorname{bc}(w)| \leq 2\lambda$. In order to obtain this result, we plug into Algorithm 1 the aforementioned conditions in the function `haveToStop` (see Algorithm 3 in the appendix).

Then, we have to adapt the function `computeDelta` to optimize the $\delta_L^{(v)}$s and the $\delta_U^{(v)}$s to the new stopping condition: in other words, we have to choose the values of $\lambda_L^{(v)}$ and $\lambda_U^{(v)}$ that should be plugged into the function `computeDelta` (we recall that the heuristic `computeDelta` chooses the $\delta_L^{(v)}$s so that we can guarantee as fast as possible that $\tilde{\boldsymbol{b}}(v) - \lambda_L^{(v)} \leq \operatorname{bc}(v) \leq \tilde{\boldsymbol{b}}(v) + \lambda_U^{(v)}$). To this purpose, we estimate the betweenness of all nodes with few samples and we sort all nodes according to these approximate values $\tilde{b}(v)$, obtaining $v_1, \dots, v_n$. The basic idea is that, for the first k nodes, we set $\lambda_U^{(v_i)} = \frac{\tilde{b}(v_{i-1}) - \tilde{b}(v_i)}{2}$, and $\lambda_L^{(v_i)} = \frac{\tilde{b}(v_i) - \tilde{b}(v_{i+1})}{2}$ (the goal is to find confidence intervals that separate the betweenness of v_i from the betweenness of v_{i+1} and v_{i-1}). For nodes that are not in the top-k, we choose $\lambda_L^{(v)} = 1$ and $\lambda_U^{(v)} = \tilde{b}(v_k) - \lambda_L^{(v_k)} - \tilde{b}(v_i)$ (the goal is to prove that v_i is not in the top-k). Finally, if $\tilde{b}(v_i) - \tilde{b}(v_{i+1})$ is small, we simply set $\lambda_L^{(v_i)} = \lambda_U^{(v_i)} = \lambda_L^{(v_{i+1})} = \lambda_U^{(v_{i+1})} = \lambda$, because we do not know if $\operatorname{bc}(v_{i+1}) > \operatorname{bc}(v_i)$, or viceversa.

6 Experimental Results

In this section, we test the four variations of our algorithm on several real-world networks, in order to evaluate their performances. The platform for our tests is a server with 1515 GB RAM and 48 Intel(R) Xeon(R) CPU E7-8857 v2 cores at 3.00GHz, running Debian GNU Linux 8. The algorithms are implemented in C++, and they are compiled using gcc 5.3.1. The source code of our algorithm is available at `https://sites.google.com/a/imtlucca.it/borassi/publications`.

Comparison with the State of the Art

The first experiment compares the performances of our algorithm KADABRA with the state of the art. The first competitor is the RK algorithm [39], available in the open-source *NetworKit* framework [44]. This algorithm uses the same estimator as our algorithm, but the stopping condition is different: it simply stops after sampling $k = \frac{c}{\epsilon^2}\left(\lfloor \log_2(\operatorname{VD} - 2)\rfloor + 1 + \log\left(\frac{1}{\delta}\right)\right)$, and it uses a heuristic to upper bound the vertex diameter. Following suggestions by the author of the *NetworKit* implementation, we set to 20 the number of samples used in the latter heuristic [7].

The second competitor is the ABRA algorithm [40], available at `http://matteo.rionda.to/software/ABRA-radebetw.tbz2`. This algorithm samples pairs of nodes (s, t), and it

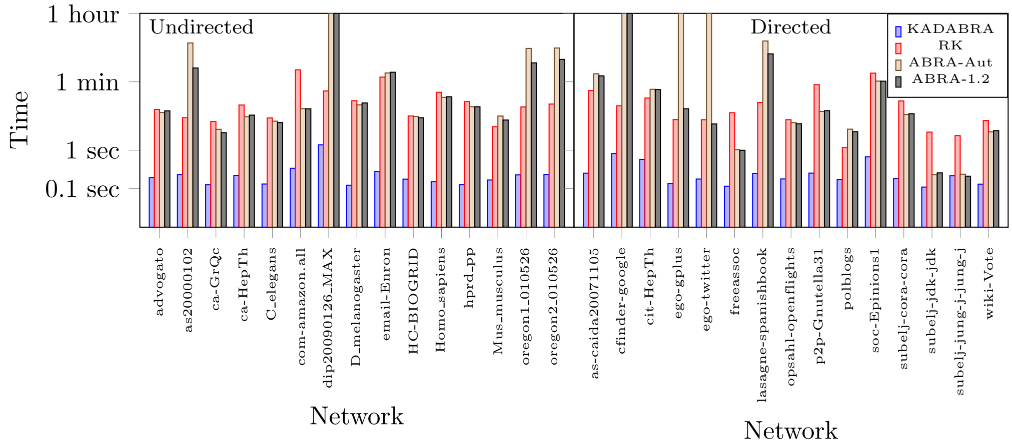

Figure 1: The time needed by the different algorithms, on all the graphs of our dataset.

adds the fraction of st-paths passing from v to the approximation of the betweenness of v, for each node v. The stopping condition is based on a key result in statistical learning theory, and there is a scheduler that decides when it should be tested. Following the suggestions by the authors, we use both the automatic scheduler ABRA-Aut, which uses a heuristic approach to decide when the stopping condition should be tested, and the geometric scheduler ABRA-1.2, which tests the stopping condition after $(1.2)^i k$ iterations, for each integer i.

The test is performed on a dataset made by 15 undirected and 15 directed real-world networks, taken from the datasets SNAP (`snap.stanford.edu/`), LASAGNE (`piluc.dsi.unifi.it/lasagne`), and KONECT (`http://konect.uni-koblenz.de/networks/`). As in [40], we have considered all values of $\lambda \in \{0.03, 0.025, 0.02, 0.015, 0.01, 0.005\}$, and $\delta = 0.1$. All the algorithms have to provide an approximation $\tilde{\boldsymbol{b}}(v)$ of $\mathrm{bc}(v)$ for each v such that $\Pr\left(\forall v, \left|\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)\right| \leq \lambda\right) \geq 1 - \delta$. In Figure 1, we report the time needed by the different algorithms on every graph for $\lambda = 0.005$ (the behavior with different values of λ is very similar). More detailed results are reported in Appendix F.

From the figure, we see that KADABRA is much faster than all the other algorithms, on all graphs: on average, our algorithm is about 100 times faster than RK in undirected graphs, and about 70 times faster in directed graphs; it is also more than 1 000 times faster than ABRA. The latter value is due to the fact that the ABRA algorithm has large running times on few networks: in some cases, it did not even conclude its computation within one hour. The authors confirmed that this behavior might be due to some bugs in the code, which seems to affect it only on specific graphs: indeed, in most networks, the performances of ABRA are better than those of the RK algorithm (but, still, not better than KADABRA).

In order to explain these data, we take a closer look at the improvements obtained through the bidirectional BFS, by considering the average number of edges m_{avg} that the algorithm visits in order to sample a shortest path (for all our competitors, $m_{\text{avg}} = m$, since they perform a full BFS). In Figure 2, for each graph in our dataset, we plot $\alpha = \frac{\log(m_{\text{avg}})}{\log(m)}$ (intuitively, this means that the average number of edges visited is m^{α}).

The figure shows that, apart from few cases, the number of edges visited is close to $n^{\frac{1}{2}}$, confirming the results in Section 4. This means that, since many of our networks have approximately 10 000 edges, the bidirectional BFS is about 100 times faster than the standard BFS. Finally, for each value of λ, we report in Figure 3 the number of samples needed by all the algorithms, averaged over all the graphs in the dataset.

From the figure, KADABRA needs to sample the smallest amount of shortest paths, and the average improvement over RK grows when λ tends to 0, from a factor 1.14 (resp.,

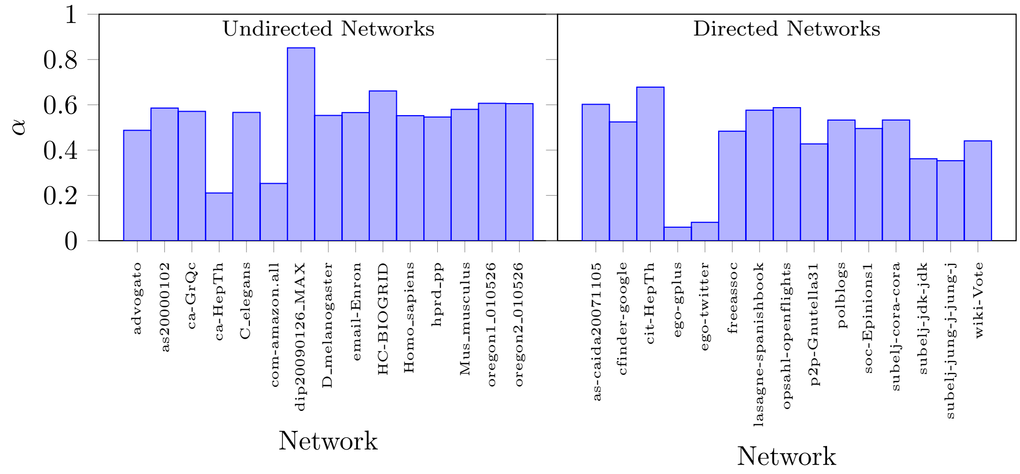

Figure 2: The exponent α such that the average number of edges visited during a bidirectional BFS is n^{α}.

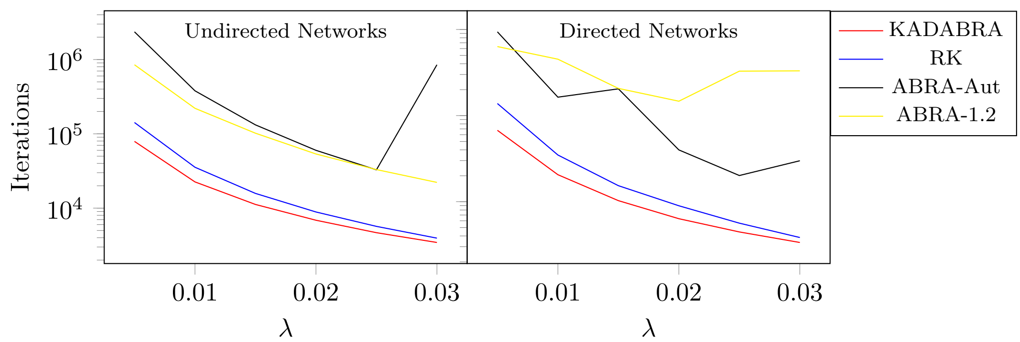

Figure 3: The average number of samples needed by the different algorithms.

1.14) if $\lambda = 0.03$, to a factor 1.79 (resp., 2.05) if $\lambda = 0.005$ in the case of undirected (resp., directed) networks. Again, the behavior of ABRA is highly influenced by the behavior on few networks, and as a consequence the average number of samples is higher. In any case, also in the graphs where ABRA has good performances, KADABRA still needs a smaller number of samples.

Computing Top-k Centralities

In the second experiment, we let KADABRA compute the top-k betweenness centralities of large graphs, which were unfeasible to handle with the previous algorithms.

The first set of graph is a series of temporal snapshots of the IMDB actor collaboration network, in which two actors are connected if they played together in a movie. The snapshots are taken every 5 years from 1940 to 2010, including a last snapshot in 2014, with 1 797 446 nodes and 145 760 312 edges. The graphs are extracted from the IMDB website (`http://www.imdb.com`), and they do not consider TV-series, awards-shows, documentaries, game-shows, news, realities and talk-shows, in accordance to what was done in `http://oracleofbacon.org`.

The other graph considered is the Wikipedia citation network, whose nodes are Wikipedia pages, and which contains an edge from page p_1 to page p_2 if the text of page p_1 contains a link to page p_2. The graph is extracted from DBPedia 3.7 (`http://wiki.dbpedia.org/`), and it consists of 4 229 697 nodes and 102 165 832 edges.

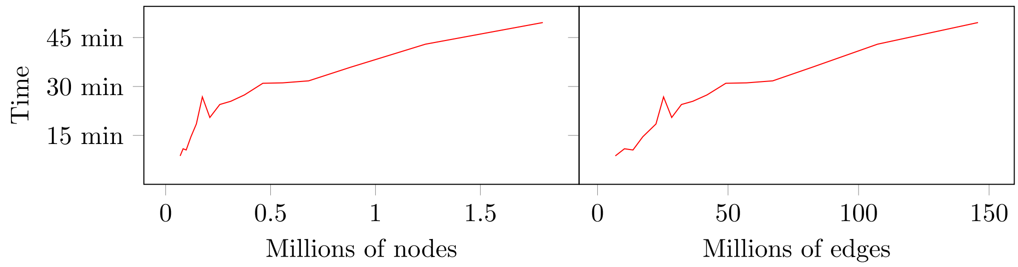

Figure 4: The total time of computation of KADABRA on increasing snapshots of the IMDB graph.

We have run our algorithm with $\lambda = 0.0002$ and $\delta = 0.1$: as discussed in Section 5, this means that either two nodes are ranked correctly, or their centrality is known with precision at most λ. As a consequence, if two nodes are not ranked correctly, the difference between their real betweenness is at most 2λ. The full results are available in Appendix G.2.

All the graphs were processed in less than one hour, apart from the Wikipedia graph, which was processed in approximately 1 hour and 38 minutes. In Figure 4, we plot the running times for the actor graphs: from the figure, it seems that the time needed by our algorithm scales slightly sublinearly with respect to the size of the graph. This result respects the results in Section 4, because the degrees in the actor collaboration network are power law distributed with exponent $\beta \approx 2.13$ (`http://konect.uni-koblenz.de/networks/actor-collaboration`). Finally, we observe that the ranking is quite precise: indeed, most of the times, there are very few nodes in the top-5 with the same ranking, and the ranking rarely contains significantly more than 10 nodes.

Acknowledgements. The authors would like to thank Matteo Riondato for several constructive comments on an earlier version of this work. We also thank Elisabetta Bergamini, Richard Lipton, and Sebastiano Vigna for helpful discussions and Holger Dell for his help with the experiments.

References

[1] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1681–1697. SIAM, 2015.

[2] Amir Abboud, Virginia V. Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–391, may 2016. URL: `http://arxiv.org/abs/1506.0179`, `arXiv:1506.0179`.

[3] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive Sampling for k-Means Clustering. In Irit Dinur, Klaus Jansen, Joseph Naor, and Jos Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, number 5687 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009.

[4] Jac M Anthonisse. The rush in a directed graph. *Stichting Mathematisch Centrum. Mathematische Besliskunde*, BN(9/71):1–10, 1971.

[5] David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. Approximating betweenness centrality. *The 5th Workshop on Algorithms and Models for the Web-Graph*, 2007.

[6] Alex Bavelas. A mathematical model for group structures. *Human organization*, 7(3):16–30, 1948.

[7] Elisabetta Bergamini. private communication, 2016.

[8] Elisabetta Bergamini, Michele Borassi, Pierluigi Crescenzi, Andrea Marino, and Henning Meyerhenke. Computing top-k closeness centrality faster in unweighted graphs. In *ALENEX*, 2016.

[9] Elisabetta Bergamini and Henning Meyerhenke. Fully-dynamic approximation of betweenness centrality. In *ESA*, 2015.

[10] Paolo Boldi, Andrea Marino, Massimo Santini, and Sebastiano Vigna. BUbiNG: Massive crawling for the masses. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web*, pages 227–228, 2014.

[11] Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980. `doi:10.1016/S0195-6698(80)80030-8`.

[12] Béla Bollobás, Svante Janson, and Oliver Riordan. The phase transition in inhomogeneous random graphs. *Random Structures and Algorithms*, 31(1):3–122, 2007.

[13] Michele Borassi, Pierluig Crescenzi, and Michel Habib. Into the square - On the complexity of some quadratic-time solvable problems. In *Proceedings of the 16th Italian Conference on Theoretical Computer Science (ICTCS)*, pages 1–17, 2015.

[14] Michele Borassi, Pierluigi Crescenzi, and Luca Trevisan. An Axiomatic and an Average-Case Analysis of Algorithms and Heuristics for Metric Properties of Graphs. *arXiv:1604.01445 [cs]*, April 2016. arXiv: 1604.01445.

[15] Stephen P. Borgatti and Martin G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28:466–484, 2006.

[16] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, jun 2001. doi:10.1080/0022250X.2001.9990249.

[17] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30:136–145, 2008.

[18] Ulrik Brandes and Christian Pich. Centrality Estimation in Large Networks. *International Journal of Bifurcation and Chaos*, 17(07):2303–2318, 2007. doi:10.1142/S0218127407018403.

[19] Bernard S Cohn and McKim Marriott. Networks and centres of integration in indian civilization. *Journal of social Research*, 1(1):1–9, 1958.

[20] Shlomi Dolev, Yuval Elovici, and Rami Puzis. Routing betweenness centrality. *J. ACM*, 57, 2010.

[21] David A. Easley and Jon M. Kleinberg. Networks, crowds, and markets - reasoning about a highly connected world. In *DAGLIB*, 2010.

[22] David Eppstein and Joseph Wang. Fast approximation of centrality. *J. Graph Algorithms Appl.*, 8:39–45, 2001.

[23] Dra Erds, Vatche Ishakian, Azer Bestavros, and Evimaria Terzi. A divide-and-conquer algorithm for betweenness centrality. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 433–441, 2015.

[24] Daniel Fernholz and Vijaya Ramachandran. The diameter of sparse random graphs. *Random Structures and Algorithms*, 31(4):482–516, 2007. doi:10.1002/rsa.

[25] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In Catherine C. McGeoch, editor, *Experimental Algorithms: 7th International Workshop, WEA 2008*, pages 319–333. Springer Berlin Heidelberg, 2008.

[26] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, dec 2001. doi:10.1006/jcss.2001.1774.

[27] Riko Jacob, Dirk Koschützki, Katharina Anna Lehmann, Leon Peeters, and Dagmar Tenfelde-Podehl. Algorithms for centrality indices. In *DAGSTUHL*, 2004.

[28] Hermann Kaindl and Gerhard Kainz. Bidirectional heuristic search reconsidered. *J. Artif. Intell. Res. (JAIR)*, 7:283–317, 1997.

[29] Yeon-sup Lim, Daniel S Menasché, Bruno Ribeiro, Don Towsley, and Prithwish Basu. Online estimating the k central nodes of a network. *Proceedings of IEEE NSW*, pages 118–122, 2011.

[30] Richard J. Lipton and Jeffrey F. Naughton. Query Size Estimation by Adaptive Sampling. *Journal of Computer and System Sciences*, 51(1):18–25, August 1995. doi:10.1006/jcss.1995.1050.

[31] Richard J. Lipton and Naughton, Jeffrey F. Estimating the size of generalized transitive closures. In *Proceedings of the 15th Int. Conf. on Very Large Data Bases*, 1989.

[32] Linyuan Lu and Fan R. K. Chung. *Complex graphs and networks*. Number no. 107 in CBMS regional conference series in mathematics. American Mathematical Society, 2006.

[33] Mark Newman. *Networks: an introduction*. OUP Oxford, 2010.

[34] Mark EJ Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical review E*, 64(1):016132, 2001.

[35] Ilkka Norros and Hannu Reittu. On a conditionally Poissonian graph process. *Advances in Applied Probability*, 38(1):59–75, 2006.

[36] Jürgen Pfeffer and Kathleen M Carley. k-centralities: local approximations of global measures based on shortest paths. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 1043–1050. ACM, 2012.

[37] Andrea Pietracaprina, Matteo Riondato, Eli Upfal, and Fabio Vandin. Mining Top-K Frequent Itemsets Through Progressive Sampling. *Data Mining and Knowledge Discovery*, 21(2):310–326, September 2010. `doi:10.1007/s10618-010-0185-7`.

[38] Ira Pohl. *Bi-directional and heuristic search in path problems.* PhD thesis, Dept. of Computer Science, Stanford University., 1969.

[39] Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2015.

[40] Matteo Riondato and Eli Upfal. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *arXiv preprint 1602.05866*, pages 1–27, 2016. `arXiv:1602.05866`.

[41] Ahmet Erdem Sariyüce, Erik Saule, Kamer Kaya, and Ümit V Çatalyürek. Shattering and compressing networks for betweenness centrality. In *SIAM Data Mining Conference (SDM). SIAM*, 2013.

[42] Marvin E Shaw. Group structure and the behavior of individuals in small groups. *The Journal of psychology*, 38(1):139–149, 1954.

[43] Alfonso Shimbel. Structural parameters of communication networks. *The bulletin of mathematical biophysics*, 15(4):501–507, 1953.

[44] Christian L. Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. Networkit: an interactive tool suite for high-performance network analysis. *arXiv preprint 1403.3005*, pages 1–25, 2014.

[45] Remco van der Hofstad. Random graphs and complex networks. Vol. II. Manuscript, 2014.

[46] Flavio Vella, Giancarlo Carbone, and Massimo Bernaschi. Algorithms and heuristics for scalable betweenness centrality computation on multi-gpu systems. *CoRR*, abs/1602.00963, 2016.

[47] Sebastiano Vigna. private communication, 2016.

[48] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[49] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1867–1877, 2014. URL: `http://epubs.siam.org/doi/abs/10.1137/1.9781611973402.135`, `doi:10.1137/1.9781611973402.135`.

A Pseudocode

Algorithm 2: the function `computeDelta`.

Input : a graph $G = (V, E)$, and two values $\lambda_L^{(v)}, \lambda_U^{(v)}$ for each $v \in V$
Output : for each $v \in V$, two values $\delta_L^{(v)}, \delta_U^{(v)}$

$\alpha \leftarrow \frac{\omega}{100}$;
$\epsilon \leftarrow 0.0001$;
foreach $i \in [1, \alpha]$ **do**
  $\boldsymbol{\pi} =$ `samplePath`();
  foreach $v \in \boldsymbol{\pi}$ **do** $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v) + 1$
end
foreach $v \in V$ **do**
  $\tilde{\boldsymbol{b}}(v) \leftarrow \tilde{\boldsymbol{b}}(v)/\alpha$;
  $c_L(v) \leftarrow \frac{2\tilde{b}(v)\omega}{(\lambda_L^{(v)})^2}$;
  $c_U(v) \leftarrow \frac{2\tilde{b}(v)\omega}{(\lambda_U^{(v)})^2}$;
end
Binary search to find C such that $\sum_{v \in V} \exp\left(-\frac{C}{c_L(v)}\right) + \exp\left(-\frac{C}{c_U(v)}\right) = \frac{\delta}{2} - \epsilon\delta$;
foreach $v \in V$ **do**
  $\delta_L^{(v)} \leftarrow \exp\left(-\frac{C}{c_L(v)}\right) + \frac{\epsilon\delta}{2n}$;
  $\delta_U^{(v)} \leftarrow \exp\left(-\frac{C}{c_U(v)}\right) + \frac{\epsilon\delta}{2n}$;
end
return $\boldsymbol{b}$;

Algorithm 3: the function `haveToStop` to compute the top-k nodes.

Input : for each node v, the values of $\tilde{\boldsymbol{b}}(v), \delta_L^{(v)}, \delta_U^{(v)}$, and the values of ω and τ
Output : True if the algorithm should stop, False otherwise

Sort nodes in decreasing order of $\tilde{\boldsymbol{b}}(v)$, obtaining $v_1, \dots, v_n$;
for $i \in [1, \dots, k]$ **do**
  if $f(\tilde{\boldsymbol{b}}(v_i), \delta_L^{(v_i)}, \omega, \tau) > \lambda$ *or* $g(\tilde{\boldsymbol{b}}(v_i), \delta_U^{(v_i)}, \omega, \tau) > \lambda$ **then**
    if $\tilde{\boldsymbol{b}}(v_{i-1}) - f(\tilde{\boldsymbol{b}}(v_{i-1}), \delta_L^{(v_{i-1})}, \omega, \tau) < \tilde{\boldsymbol{b}}(v_i) + g(\tilde{\boldsymbol{b}}(v_i), \delta_U^{(v_i)}, \omega, \tau)$ *or* $\tilde{\boldsymbol{b}}(v_i) - f(\tilde{\boldsymbol{b}}(v_i), \delta_L^{(v_i)}, \omega, \tau) < \tilde{\boldsymbol{b}}(v_{i+1}) + g(\tilde{\boldsymbol{b}}(v_{i+1}), \delta_U^{(v_{i+1})}, \omega, \tau)$ **then**
      return *False*;
    end
  end
end
for $i \in [k+1, \dots, n]$ **do**
  if $f(\tilde{\boldsymbol{b}}(v_i), \delta_L^{(v_i)}, \omega, \tau) > \lambda$ *or* $g(\tilde{\boldsymbol{b}}(v_i), \delta_U^{(v_i)}, \omega, \tau) > \lambda$ **then**
    if $\tilde{\boldsymbol{b}}(v_k) - f(\tilde{\boldsymbol{b}}(v_k), \delta_L^{(v_k)}, \omega, \tau) < \tilde{\boldsymbol{b}}(v_i) + g(\tilde{\boldsymbol{b}}(v_i), \delta_U^{(v_i)}, \omega, \tau)$ **then**
      return *False*;
    end
  end
end
return *True*;

B Concentration Inequalities

Lemma 5 (Hoeffding's inequality)**.** *Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_k$ be independent random variables such that $a_i < \boldsymbol{X}_i < b_i$, and let $\boldsymbol{X} = \sum_{i=1}^k \boldsymbol{X}_i$. Then,*

$$\Pr\left(|\boldsymbol{X} - \mathbb{E}[X]| \geq \lambda\right) \leq \exp\left\{-\frac{2\lambda^2}{\sum_{i=1}^k (b_i - a_i)^2}\right\}.$$

Remark 6. *If we apply Hoeffding's inequality with $\boldsymbol{X}_i = X_v^{\boldsymbol{\pi}}$, $\boldsymbol{X} = k\boldsymbol{b}(v) = \sum_{i=1}^k X_v^{\boldsymbol{\pi}}$, $a_i = 0$, $b_i = 1$, we obtain that* $\Pr\left(|\boldsymbol{b}(v) - \mathrm{bc}(v)| > \lambda\right) < 2e^{-2k\lambda^2}$. *Then, if we fix* $\delta = 2e^{-2k\lambda^2}$*,* *the error is* $\lambda = \sqrt{\frac{\log(2/\delta)}{2k}}$, *and the minimum* k *needed to obtain an error* λ *on the betweenness of a single node is* $\frac{1}{2\lambda^2}\log(2/\delta)$.

Lemma 7 (Chernoff bound ([32]))**.** *Let* $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$ *be independent random variables such that* $\boldsymbol{X}_i \leq M$ *for each* $1 \leq i \leq n$, *and let* $\boldsymbol{X} = \sum_{i=1}^{k} \boldsymbol{X}_i$. *Then,*

$$\Pr\left(\boldsymbol{X} \geq \mathbb{E}[\boldsymbol{X}] + \lambda\right) \leq \exp\left\{-\frac{\lambda^2}{2(\sum_{i=1}^{n}\mathbb{E}[\boldsymbol{X}_i^2] + M\lambda/3)}\right\}.$$

Theorem 8 (McDiarmid '98 ([32]))**.** *Let* X *be a martingale associated with a filter* $\mathcal{F}$, *satisfying*

- $\mathrm{Var}\left(X_i | \ \mathcal{F}_i\right) \leq \sigma_i$ *for* $1 \leq i \leq \ell$,
- $|X_i - X_{i-1}| \leq M$, *for* $1 \leq i \leq \ell$.

Then, we have

$$\Pr\left(X - \mathbb{E}\left(X\right) \geq \lambda\right) \leq \exp\left(-\frac{\lambda^2}{2\left(\sum_{i=1}^{\ell}\sigma_i^2 + M\lambda/3\right)}\right).$$

C Proof of Theorem 2

In our algorithm, we sample $\boldsymbol{\tau}$ shortest paths $\boldsymbol{\pi}_i$, where $\boldsymbol{\tau}$ is a random variable such that $\boldsymbol{\tau} = \tau$ can be decided by looking at the first τ paths sampled (see Algorithm 1). Furthermore, thanks to Eq. (3) in [39], we assume that $\boldsymbol{\tau} \leq \omega$ for some fixed $\omega \in \mathbb{R}^+$ such that, after ω steps, $\Pr(\forall v, |\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| \leq \lambda) \geq 1 - \frac{\delta}{2}$. When the algorithm stops, our estimate of the betweenness is $\tilde{\boldsymbol{b}}(v) := \frac{1}{\boldsymbol{\tau}}\sum_{i=1}^{\boldsymbol{\tau}} \boldsymbol{X}_i(v)$, where $\boldsymbol{X}_i(v)$ is 1 if v belongs to $\boldsymbol{\pi}_i$, 0 otherwise.

To estimate the error, we use the following theorem.

Theorem 9. *For each node* v *and for every fixed real numbers* δ_L, δ_U, *it holds*

$$\Pr\left(\mathrm{bc}(v) \leq \tilde{\boldsymbol{b}}(v) - f\left(\tilde{\boldsymbol{b}}(v), \delta_L, \omega, \boldsymbol{\tau}\right)\right) \leq \delta_L \quad \text{and}$$
$$\Pr\left(\mathrm{bc}(v) \geq \tilde{\boldsymbol{b}}(v) + g\left(\tilde{\boldsymbol{b}}(v), \delta_U, \omega, \boldsymbol{\tau}\right)\right) \leq \delta_U,$$

where

$$f\left(\tilde{\boldsymbol{b}}(v), \delta_L, \omega, \boldsymbol{\tau}\right) = \frac{1}{\boldsymbol{\tau}}\log\frac{1}{\delta_L}\left(\frac{1}{3} - \frac{\omega}{\boldsymbol{\tau}} + \sqrt{\left(\frac{1}{3} - \frac{\omega}{\boldsymbol{\tau}}\right)^2 + \frac{2\tilde{\boldsymbol{b}}(v)\omega}{\log\frac{1}{\delta_L}}}\right) \quad \text{and} \tag{5}$$

$$g\left(\tilde{\boldsymbol{b}}(v), \delta_U, \omega, \boldsymbol{\tau}\right) = \frac{1}{\boldsymbol{\tau}}\log\frac{1}{\delta_U}\left(\frac{1}{3} + \frac{\omega}{\boldsymbol{\tau}} + \sqrt{\left(\frac{1}{3} + \frac{\omega}{\boldsymbol{\tau}}\right)^2 + \frac{2\tilde{\boldsymbol{b}}(v)\omega}{\log\frac{1}{\delta_U}}}\right). \tag{6}$$

We prove Theorem 9 in Section C.1. In the rest of this section, we show how this theorem implies Theorem 2. To simplify notation, we often omit the arguments of the function f and g.

Proof of Theorem 2. Let $\boldsymbol{E}_1$ be the event $(\boldsymbol{\tau} = \omega \wedge \exists v \in V, |\tilde{\boldsymbol{b}}(v) - \mathrm{bc}(v)| > \lambda)$, and let $\boldsymbol{E}_2$ be the event $(\boldsymbol{\tau} < \omega \wedge (\exists v \in V, -f \geq \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \vee \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \geq g))$. Let us also denote $\tilde{\boldsymbol{b}}_\tau(v) = \frac{1}{\tau}\sum_{i=1}^{\tau}\boldsymbol{X}_i(v)$ (note that $\tilde{\boldsymbol{b}}_{\boldsymbol{\tau}}(v) = \tilde{\boldsymbol{b}}(v)$).

By our choice of ω and Eq. (3) in [39],

$$\Pr(\boldsymbol{E}_1) \leq \Pr(\exists v \in V, |\tilde{\boldsymbol{b}}_\omega(v) - \mathrm{bc}(v)| > \lambda) \leq \frac{\delta}{2}$$

where $\tilde{\boldsymbol{b}}_\omega(v)$ is the approximate betweenness of v after ω samples. Furthermore, by Theorem 9,

$$\begin{aligned}\Pr(\boldsymbol{E}_2) &\leq \sum_{v \in V} \Pr(\boldsymbol{\tau} < \omega \wedge -f \geq \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v)) + \Pr(\boldsymbol{\tau} < \omega \wedge \mathrm{bc}(v) - \tilde{\boldsymbol{b}}(v) \leq g) \\ &\leq \sum_{v \in V} \delta_L^{(v)} + \delta_U^{(v)} \leq \frac{\delta}{2}.\end{aligned}$$

By a union bound, $\Pr(\boldsymbol{E}_1 \vee \boldsymbol{E}_2) \leq \Pr(\boldsymbol{E}_1) + \Pr(\boldsymbol{E}_1) \leq \delta$, concluding the proof of Theorem 2. $\square$

C.1 Proof of Theorem 9

Since this theorem deals with a single node v, let us simply write $\mathrm{bc} = \mathrm{bc}(v), \tilde{\boldsymbol{b}} = \tilde{\boldsymbol{b}}(v), \boldsymbol{X}_i = \boldsymbol{X}_i(v)$. Let us consider $\boldsymbol{Y}^\tau = \sum_{i=1}^{\tau} (\boldsymbol{X}_i - \mathrm{bc})$ (we recall that $\boldsymbol{X}_i = 1$ if v is in the i-th path sampled, $\boldsymbol{X}_i = 0$ otherwise). Clearly, $\boldsymbol{Y}^\tau$ is a martingale, and $\boldsymbol{\tau}$ is a stopping time for $\boldsymbol{Y}^\tau$: this means that also $\boldsymbol{Z}^\tau = \boldsymbol{Y}^{\min(\boldsymbol{\tau}, \tau)}$ is a martingale.

Let us apply Theorem 8 to the martingales $\boldsymbol{Z}$ and $-\boldsymbol{Z}$: for each fixed $\lambda_L, \lambda_U > 0$ we have

$$\Pr\left(\boldsymbol{Z}^\omega \geq \lambda_L\right) = \Pr\left(\boldsymbol{\tau}\tilde{\boldsymbol{b}} - \boldsymbol{\tau}\,\mathrm{bc} \geq \lambda_L\right) \leq \exp\left(-\frac{\lambda_L^2}{2\left(\omega\,\mathrm{bc} + \lambda_L/3\right)}\right) = \delta_L \quad \text{and} \tag{7}$$

$$\Pr\left(-\boldsymbol{Z}^\omega \geq \lambda_U\right) = \Pr\left(\boldsymbol{\tau}\tilde{\boldsymbol{b}} - \boldsymbol{\tau}\,\mathrm{bc} \leq -\lambda_U\right) \leq \exp\left(-\frac{\lambda_U^2}{2\left(\omega\,\mathrm{bc} + \lambda_U/3\right)}\right) = \delta_U. \tag{8}$$

We now show how to prove (5) from (7). The way to derive (6) from (8) is analogous.

If we express λ_L as a function of δ_L we get

$$\lambda_L^2 = 2\log\frac{1}{\delta_L}\left(\omega\,\mathrm{bc} + \frac{\lambda_L}{3}\right) \iff \lambda_L^2 - \frac{2}{3}\lambda_L \log\frac{1}{\delta_L} - 2\omega\,\mathrm{bc}\log\frac{1}{\delta_L} = 0,$$

which implies that

$$\lambda_L = \frac{1}{3}\log\frac{1}{\delta_L} \pm \sqrt{\frac{1}{9}\left(\log\frac{1}{\delta_L}\right)^2 + 2\omega\,\mathrm{bc}\log\frac{1}{\delta_L}}.$$

Since (7) holds for any positive value λ_L, it also holds for the value corresponding to the positive solution of this equation, that is,

$$\lambda_L = \frac{1}{3}\log\frac{1}{\delta_L} + \sqrt{\frac{1}{9}\left(\log\frac{1}{\delta_L}\right)^2 + 2\omega\,\mathrm{bc}\log\frac{1}{\delta_L}}.$$

Plugging this value into (7), we obtain

$$\Pr\left(\boldsymbol{\tau}\tilde{\boldsymbol{b}} - \boldsymbol{\tau}\,\mathrm{bc} \geq \frac{1}{3}\log\frac{1}{\delta_L} + \sqrt{\frac{1}{9}\left(\log\frac{1}{\delta_L}\right)^2 + 2\omega\,\mathrm{bc}\log\frac{1}{\delta_L}}\right) \leq \delta_L. \tag{9}$$

By assuming $\tilde{\boldsymbol{b}} - \mathrm{bc} \geq \frac{1}{3\boldsymbol{\tau}}\log(\frac{1}{\delta_L})$, the event in (9) can be rewritten as

$$(\boldsymbol{\tau}\,\mathrm{bc})^2 - 2\,\mathrm{bc}\left(\boldsymbol{\tau}^2\tilde{\boldsymbol{b}} + \omega\log\frac{1}{\delta_L} - \frac{1}{3}\boldsymbol{\tau}\log\frac{1}{\delta_L}\right) - \frac{2}{3}\log\frac{1}{\delta_L}\boldsymbol{\tau}\tilde{\boldsymbol{b}} + \left(\boldsymbol{\tau}\tilde{\boldsymbol{b}}\right)^2 \geq 0.$$

By solving the previous quadratic equation w.r.t. bc we get

$$\mathrm{bc} \leq \tilde{\boldsymbol{b}} + \log\frac{1}{\delta_L}\left(\frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}} - \sqrt{\left(\frac{\tilde{\boldsymbol{b}}}{\log\frac{1}{\delta_L}} + \frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}}\right)^2 - \left(\frac{\tilde{\boldsymbol{b}}}{\log\frac{1}{\delta_L}}\right)^2 + \frac{2}{3\boldsymbol{\tau}}\frac{\tilde{\boldsymbol{b}}}{\log\frac{1}{\delta_L}}}\right),$$

where we only considered the solution which upper bounds bc, since we assumed $\tilde{\boldsymbol{b}} - \mathrm{bc} \geq \frac{1}{3\tau}\log(\frac{1}{\delta_L})$. After simplifying the terms under the square root in the previous expression, we get

$$\mathrm{bc} \leq \tilde{\boldsymbol{b}} + \log\frac{1}{\delta_L}\left(\frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}} - \sqrt{\left(\frac{\omega}{\boldsymbol{\tau}^2} - \frac{1}{3\boldsymbol{\tau}}\right)^2 + \frac{2\tilde{\boldsymbol{b}}\omega}{\boldsymbol{\tau}^2\log\frac{1}{\delta_L}}}\right),$$

which means that

$$\Pr\left(\mathrm{bc} \leq \tilde{\boldsymbol{b}} - f\left(\tilde{\boldsymbol{b}}, \delta_L, \omega, \boldsymbol{\tau}\right)\right) \leq \delta_L,$$

concluding the proof.

D How to Choose $\delta_L^{(v)}, \delta_U^{(v)}$

In Appendix C, we proved that our algorithm works for any choice of the values $\delta_L^{(v)}, \delta_U^{(v)}$. In this section, we show how we can heuristically compute such values, in order to obtain the best performances.

For each node v, let $\lambda_L^{(v)}, \lambda_U^{(v)}$ be the lower and the upper maximum error that we want to obtain on the betweenness of v: if we simply want all errors to be smaller than λ, we choose $\lambda_L^{(v)}, \lambda_U^{(v)} = \lambda$, but for other purposes different values might be needed. We want to minimize the time τ such that the approximation of the betweenness at time τ is in the confidence interval required. In formula, we want to minimize

$$\min\left\{\tau \in \mathbb{N} : \forall v \in V, \left(f\left(\tilde{\boldsymbol{b}}_\tau(v), \delta_L^{(v)}, \omega, \tau\right) \leq \lambda_L^{(v)} \wedge g\left(\tilde{\boldsymbol{b}}_\tau(v), \delta_U^{(v)}, \omega, \tau\right) \leq \lambda_U^{(v)}\right)\right\} \tag{10}$$

where $\tilde{\boldsymbol{b}}_\tau(v)$ is the approximation of $\mathrm{bc}(v)$ obtained at time τ, and

$$f\left(\tau, \tilde{\boldsymbol{b}}_\tau, \delta_L, \omega\right) = \frac{1}{\tau}\log\frac{1}{\delta_L}\left(\frac{1}{3} - \frac{\omega}{\tau} + \sqrt{\left(\frac{1}{3} - \frac{\omega}{\tau}\right)^2 + \frac{2\tilde{\boldsymbol{b}}_\tau\omega}{\log\frac{1}{\delta_L}}}\right) \quad \text{and}$$

$$g\left(\tau, \tilde{\boldsymbol{b}}_\tau, \delta_U, \omega\right) = \frac{1}{\tau}\log\frac{1}{\delta_U}\left(\frac{1}{3} + \frac{\omega}{\tau} + \sqrt{\left(\frac{1}{3} + \frac{\omega}{\tau}\right)^2 + \frac{2\tilde{\boldsymbol{b}}_\tau\omega}{\log\frac{1}{\delta_U}}}\right).$$

The goal of this section is to provide deterministic values of $\delta_L^{(v)}, \delta_U^{(v)}$ that minimize the value in (10), and such that $\sum_{v\in V}\delta_L^{(v)} + \delta_U^{(v)} < \frac{\delta}{2}$. To obtain our estimate, we replace $\tilde{\boldsymbol{b}}_\tau(v)$ with an approximation $\tilde{b}(v)$, that we compute by sampling α paths, before starting the algorithm (in our code, $\alpha = \frac{\omega}{100}$). Furthermore, we consider a simplified version of (10): in most cases, λ_L is much smaller than all other quantities in play, and since ω is proportional to $\frac{1}{\lambda_L^2}$, we can safely assume $f(\tau, \tilde{b}(v), \delta_L^{(v)}, \omega) \approx \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\delta_L}}$ and $g(\tau, \tilde{b}(v), \delta_U^{(v)}, \omega) \approx \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\delta_U}}$. Hence, in place of the value in (10), our heuristic tries to minimize

$$\min\left\{\tau \in \mathbb{N} : \forall v \in V, \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\delta_L^{(v)}}} \leq \lambda_L^{(v)} \wedge \sqrt{\frac{2\tilde{b}(v)\omega}{\tau^2}\log\frac{1}{\delta_U^{(v)}}} \leq \lambda_U^{(v)}\right\}.$$

Solving with respect to τ, we are trying to minimize

$$\max_{v\in V}\left(\max\left(\sqrt{\frac{2\tilde{b}(v)\omega}{\left(\lambda_L^{(v)}\right)^2}\log\frac{1}{\delta_L^{(v)}}}, \sqrt{\frac{2\tilde{b}(v)\omega}{\left(\lambda_U^{(v)}\right)^2}\log\frac{1}{\delta_U^{(v)}}}\right)\right).$$

which is the same as minimizing $\max_{v\in V} \max\left(c_L(v)\log\frac{1}{\delta_L^{(v)}}, c_U(v)\log\frac{1}{\delta_U^{(v)}}\right)$ for some constants $c_L(v), c_U(v)$, conditioned on $\sum_{v\in V}\delta_L^{(v)}+\delta_U^{(v)}<\frac{\delta}{2}$. We claim that, among the possible choices of $\delta_L^{(v)}, \delta_U^{(v)}$, the best choice makes all the terms in the maximum equal: otherwise, if two terms were different, we would be able to slightly increase and decrease the corresponding values, in order to decrease the maximum. This means that, for some constant C, for each v, $c_L(v)\log\frac{1}{\delta_L^{(v)}}=c_U(v)\log\frac{1}{\delta_L^{(v)}}=C$, that is, $\delta_L^{(v)}=\exp(-\frac{C}{c_L(v)})$, $\delta_U^{(v)}=\exp(-\frac{C}{c_U(v)})$. In order to find the largest constant C such that $\sum_{v\in V}\delta_L^{(v)}+\delta_U^{(v)}\leq\frac{\delta}{2}$, we use a binary search procedure on all possible constants C.

Finally, if $c_L(v)=0$ or $c_U(v)=0$, this procedure chooses $\delta_L^{(v)}=0$: to avoid this problem, we impose $\sum_{v\in V}\delta_L^{(v)}+\delta_U^{(v)}\leq\frac{\delta}{2}-\epsilon\delta$, and we add $\frac{\epsilon\delta}{2n}$ to all the $\delta_L^{(v)}$s and all the $\delta_U^{(v)}$s (in our code, we choose $\epsilon=0.001$). The pseudocode of the algorithm is available in Algorithm 2.

E Balanced Bidirectional BFS on Random Graphs

In this appendix, we formally prove that the bidirectional BFS is efficient in several models of random graphs: the Configuration Model (CM, [11]), and Rank-1 Inhomogeneous Random Graph models (IRG, [45, Chapter 3]), such as the Chung-Lu model [32], the Norros-Reittu model [35], and the Generalized Random Graph [45, Chapter 3]. All these models are defined by fixing the number n of nodes and n weights ρ_v, and by creating edges at random, in a way that node v gets degree close to ρ_v.

More formally, the edges are generated as follows:

- in the CM, each node is associated to ρ_v half-edges, or stubs; edges are created by randomly pairing these $M=\sum_{v\in V}\rho_v$ stubs (we assume the number of stubs to be even, by adding a stub to a random node if necessary).
- in IRG, an edge between a node v and a node w exists with probability $f(\frac{\rho_v\rho_w}{M})$, where $M=\sum_{v\in V}\rho_v$, and the existence of different edges is independent. Different choices of the function f create different models.
  - In general, we assume that f satisfies the following conditions:
    - f is derivable at least twice in 0;
    - f is increasing;
    - $f'(0)=1$;
  - in the Chung-Lu model, $f(x)=\min(x,1)$;
  - in the Norros-Reittu model, $f(x)=1-e^{-x}$;
  - in the Generalized Random Graph model, $f(x)=\frac{x}{1+x}$.

It remains to define how we choose the weights ρ_v, when the number of nodes n tends to infinity. In the line of previous works [35, 24, 45], we consider a sequence of graphs G_i, whose number of nodes n_i tends to infinity, and whose degree distribution λ_i satisfy the following:

1. there is a probability distribution λ such that the λ_is tend to λ in distribution;
2. $M_1(\lambda_i)$ tends to $M_1(\lambda)<\infty$, where $M_1(\lambda)$ is the first moment of λ;
3. one of the following two conditions hold:
   (a) $M_2(\lambda_i)$ tends to $M_2(\lambda)<\infty$, where $M_2(\lambda)$ is the second moment of λ;
   (b) λ is a power law distribution with $2<\beta<3$, and there is a global constant C such that, for each d, $\Pr(\lambda_i\geq d)\leq\frac{C}{d^{\beta-1}}$.

For example, these assumptions are satisfied with probability 1 if we choose the degrees independently, according to a distribution λ with finite mean [45, Section 6.1,7.2].

Remark 10. *Note that an aspect often neglected in previous work when it comes to computing shortest paths is the fact that the number of shortest paths between a pair of nodes may be exponential, thus requiring to work with a linear number of bits. While real-world complex networks are typically sparse with logarithmic diameter, in order to avoid such issue it is sufficient to assume that addition and comparison require constant time.*

Remark 11. *These assumptions cover the Erdös-Renyi random graph with constant average degree, and all power law distributions with $\beta > 2$ (because, if $\beta > 3$, then $M_2(\lambda)$ is finite).*

Remark 12. *Assumption 3b seems less natural than the other assumptions. However, it is necessary to exclude pathological cases: for example, assume that G_i has $n-2$ nodes chosen according to a power law distribution, and 2 nodes u, v with weight $n^{1-\epsilon}$. All assumption except 3b are satisfied, but the bidirectional BFS is not efficient, because if s is a neighbor of u with degree 1, and t is a neighbor of v with degree 1, then a bidirectional BFS from s and t needs to visit all neighbors of u or all neighbors of v, and the time needed is $\Omega(n^{1-\epsilon})$.*

We say that a random graph has a property π asymptotically almost surely (a.a.s.) if $\Pr(\pi(G_i))$ tends to 1 when n tends to infinity. We say that a random graph has a property π with high probability (w.h.p.) if $\frac{\Pr(\pi(G_i))}{n_i^k}$ tends to 0 for each $k > 0$.

Before proving the main theorem, we need two more definitions and a technical assumption.

Definition 13. *In the CM, let $\rho_{\text{res}} = \frac{M_2(\lambda)}{M_1(\lambda)} - 1$. In IRG, let $\rho_{\text{res}} = \frac{M_2(\lambda)}{M_1(\lambda)}$ (if λ is a power law distribution with $2 < \beta < 3$, we simply define $\rho_{\text{res}} = +\infty$).*

Definition 14. *Given a set $V' \subseteq V$, the volume of V' is $\rho_{V'} = \sum_{v \in V'} \rho_v$. Furthermore, if $V' = \mathbf{\Gamma}^d(s)$, we abbreviate $\rho_{\mathbf{\Gamma}^d(s)}$ with $\boldsymbol{r}^l(s)$.*

The value ρ_{res} is closely related to $\frac{\boldsymbol{r}^{l+1}(s)}{\boldsymbol{r}^l(s)}$: informally, the expected value of this fraction is ρ_{res}. For this reason, if $\rho_{\text{res}} < 1$, then the size of neighbors tends to decrease, and all connected components have $\mathcal{O}(\log n)$ nodes. Conversely, if $\rho_{\text{res}} > 1$, then the size of neighbors tends to increase, and there is a *giant component* of size $\Theta(n)$ (for a proof of these facts, see [45, Section 2.3 and Chapter 4]). Our last assumption is that $\rho_{\text{res}} > 1$, in order to ensure the existence of the giant component.

Under these assumptions, we prove Theorem 4, following the sketch in Section 4. We start by linking the degrees and the weights of nodes.

Lemma 15. *For each node v, $\rho_v n^{-\epsilon} \leq \deg(v) \leq \rho_v n^{\epsilon}$ w.h.p..*

Proof. We use [14, Lemmas 32 and 37][2]: these lemmas imply that, for each $\epsilon > 0$, if $\rho_v > n^{\epsilon}$, $(1-\epsilon)\rho_v \leq \deg(v) \leq (1+\epsilon)\rho_v$ w.h.p.. We have to handle the case where $\rho_v < n^{\epsilon}$: one of the two inequalities is empty, while for the other inequality we observe that, if we decrease the weight of v, the degree of v can only decrease. Hence, if $\rho_v < n^{\epsilon}$, $\deg(v) < (1+\epsilon)n^{\epsilon}$, and the result follows by changing the value of ϵ. □

Following the intuitive proof, we have linked the number of visited edges with their weights. Let us define an abbreviation for the volume of the nodes at distance l from s.

Definition 16. *We denote by $\boldsymbol{r}^l(s)$ the volume of nodes at distance exactly l from s. In the CM, we denote by $\boldsymbol{R}^l(s)$ the set of stubs at distance l from s.*

Now, we need to show that, if $\boldsymbol{r}^{l_s}(s), \boldsymbol{r}^{l_t}(t) > n^{\frac{1}{2}+\epsilon}$, then $d(s,t) \leq l_s + l_t + 2$ w.h.p..

[2] This paper uses a further assumption on IRG, but the proofs of Lemmas 32 and 39 do not rely on this assumption.

Lemma 17. *Assume that $\boldsymbol{r}^{l_s}(s) > n^{\frac{1}{2}+\epsilon}$, $\boldsymbol{r}^{l_t}(t) > n^{\frac{1}{2}+\epsilon}$, and $\boldsymbol{r}^{l_s-1}(s), \boldsymbol{r}^{l_t-1}(t) < (1-\epsilon)n^{\frac{1}{2}+\epsilon}$. Then, $d(s,t) \leq l_s + l_t + 2$.*

Proof. Let us assume that we know the structure of $\boldsymbol{N}^{l_s}(s)$ and $\boldsymbol{N}^{l_t}(t)$, that is, for each possible structure S of the subgraph induced by all nodes at distance l_s from s and distance l_t from t, let E_S be the event that $\boldsymbol{N}^{l_s}(s)$ and $\boldsymbol{N}^{l_t}(t)$ are exactly S. If we prove that $\Pr(d(s,t) \leq l + l' + 2 | E_S) < \epsilon$, then $\Pr(\boldsymbol{r}^{l+1}(s) > \boldsymbol{r}^{l}(s)) = \sum_S \Pr(\boldsymbol{r}^{l+1}(s) > \boldsymbol{r}^{l}(s) | E_S) \Pr(E_S) < \sum_S \epsilon \Pr(E_S) = \epsilon$. First of all, if S is such that the two neighborhoods touch each other, $\Pr(d(s,t) \leq l + l' + 2 | E_S) = 0 < \epsilon$. Otherwise, we consider separately the CM and IRG.

In the CM, conditioned on E_S, the stubs that are paired with stubs in $\boldsymbol{R}^{l_s}(s)$ are a random subset of the set of stubs that are not paired in S. This random subset has size at least $\epsilon n^{\frac{1}{2}+\epsilon} \geq n^{\frac{1+\epsilon}{2}}$ (because ϵ is a fixed constant, and n tends to infinity). Since the total number of stubs is $\mathcal{O}(n)$, and since the number of stubs in $\boldsymbol{R}^{l_t}(t)$ is at least $\epsilon n^{\frac{1+\epsilon}{2}}$, one of the stubs in $\boldsymbol{R}^{l_t}(t)$ is paired with a stub in $\boldsymbol{r}^{l_s}(s)$ w.h.p., and $d(s,t) \leq l_s + l_t + 1$.

In IRG, the probability that a node v is not connected to any node in $\boldsymbol{\Gamma}^{l_s}(s)$ is at most $\prod_{w \in \boldsymbol{\Gamma}^{l_s}(s)} (1 - f(\frac{\rho_v \rho_w}{M})) = \prod_{w \in \boldsymbol{\Gamma}^{l_s}(s)} (1 - \Omega(\frac{\rho_w}{M})) = \exp(-\sum_{w \in \boldsymbol{\Gamma}^{l_s}(s)} \Omega(\frac{\rho_w}{M})) = \exp(-\Omega(\frac{\boldsymbol{r}^{l_s}(s)}{M})) = 1 - \Omega(\frac{\boldsymbol{r}^{l_s}(s)}{M}) = 1 - \Omega(n^{-\frac{1}{2}+\epsilon})$. This means that v belongs to $\boldsymbol{\Gamma}^{l_s+1}(s)$ with probability $\Omega(n^{-\frac{1}{2}+\epsilon})$, and similarly it belongs to $\boldsymbol{\Gamma}^{l_t+1}(t)$ with probability $\Omega(n^{-\frac{1}{2}+\epsilon})$. Since the two events are independent, the probability that v belongs to both is $\Omega(n^{-1+2\epsilon})$. Since, for each node v, the events that v belongs to $\boldsymbol{\Gamma}^{l_s+1}(s) \cap \boldsymbol{\Gamma}^{l_t+1}(t)$ are independent, by a straightforward application of Hoeffding's inequality, w.h.p., there is a node v that belongs to $\boldsymbol{\Gamma}^{l_s+1}(s) \cap \boldsymbol{\Gamma}^{l_t+1}(t)$, and $d(s,t) \leq l_s + l_t + 2$ w.h.p., concluding the proof. ☐

The next ingredient is used to bound the first integers l_s, l_t such that $\boldsymbol{r}^{l_s}(s), \boldsymbol{r}^{l_t}(t) > n^{\frac{1}{2}+\epsilon}$.

Theorem 18 (Theorem 5.1 in [24] for the CM, Theorem 14.8 in [12] for IRG (see also [45, 14]))**.** *The diameter of a graph generated through the aforementioned models is $\mathcal{O}(\log n)$.*

The last ingredient of our proof is an upper bound on the size of $\boldsymbol{r}^{l_s}(s)$ and $\boldsymbol{r}^{l_t}(t)$.

Lemma 19. *With high probability, for each $s \in V$ and for each l such that $\sum_{i=0}^{l} \boldsymbol{r}^{l}(s) < n^{\frac{1}{2}+\epsilon}$, $\boldsymbol{r}^{l+1}(s) < n^{\frac{1}{2}+3\epsilon}$ if λ has finite second moment, $\boldsymbol{r}^{l+1}(s) < n^{\frac{4-\beta}{2}+3\epsilon}$ if λ is power law with $2 < \beta < 3$.*

Proof. We consider separately nodes with weight at most $n^{\frac{1}{2}-2\epsilon}$ from nodes with bigger weights: in the former case, we bound the number of such nodes that are in $\boldsymbol{R}^{l+1}(s)$, while in the latter case we bound the total number of nodes with weight at least $n^{\frac{1}{2}-2\epsilon}$. Let us start with nodes with the latter case.

Claim: for each ϵ, $\sum_{\rho_v \geq n^{\frac{1}{2}-\epsilon}} \rho_v$ is smaller than $n^{\frac{1}{2}+3\epsilon}$ if λ has finite second moment, and it is smaller than $n^{\frac{4-\beta}{2}+3\epsilon}$ if λ is power law with $2 < \beta < 3$.

Proof of claim. If λ has finite second moment, by Chebyshev inequality, for each α,

$$\Pr\left(\lambda_i > n^{\frac{1}{2}+\alpha}\right) \leq \frac{\mathrm{Var}(\lambda_i)}{n^{1+2\alpha}} \leq \frac{M_2(\lambda_i)}{n^{1+2\alpha}} = \mathcal{O}\left(\frac{M_2(\lambda)}{n^{1+2\alpha}}\right) = \mathcal{O}\left(n^{-1-2\alpha}\right).$$

For $\alpha = \epsilon$, this means that no node has weight bigger than $n^{\frac{1}{2}+\epsilon}$, and for $\alpha = -\epsilon$, this means that the number of nodes with weight bigger than $n^{\frac{1}{2}-\epsilon}$ is at most $n^{2\epsilon}$. We conclude that $\sum_{\rho_v \geq n^{\frac{1}{2}-\epsilon}} \rho_v \leq \sum_{\rho_v \geq n^{\frac{1}{2}-\epsilon}} n^{\frac{1}{2}+\epsilon} \leq n^{\frac{1}{2}+3\epsilon}$.

If λ is power law with $2 < \beta < 3$, by Assumption 3b the number of nodes with weight at least d is at most $Cnd^{-\beta+1}$. Consequently, using Abel's summation technique,

$$\begin{aligned}
\sum_{\rho_v \geq n^{\frac{1}{2}-\epsilon}} \rho_v &= \sum_{d=\rho_v}^{+\infty} d|\{v : \rho_v = d\}| \\
&= \sum_{d=n^{\frac{1}{2}-\epsilon}}^{+\infty} d(|\{v : \rho_v \geq d\}| - |\{v : \rho_v \geq d+1\}|) \\
&= \sum_{d=n^{\frac{1}{2}-\epsilon}}^{+\infty} d|\{v : \rho_v \geq d\}| - \sum_{d=n^{\frac{1}{2}-\epsilon}+1}^{+\infty} (d-1)|\{v : \rho_v \geq d\}| \\
&= n^{\frac{1}{2}-\epsilon}|\{v : \rho_v \geq n^{\frac{1}{2}-\epsilon}\}| + \sum_{d=n^{\frac{1}{2}-\epsilon}+1}^{+\infty} |\{v : \rho_v \geq d\}| \\
&\leq Cn^{\frac{1}{2}-\epsilon} n^{1-\left(\frac{1}{2}-\epsilon\right)(\beta-1)} + \sum_{d=n^{\frac{1}{2}-\epsilon}+1}^{+\infty} Cnd^{-\beta+1} \\
&= \mathcal{O}\left(n^{\frac{4-\beta}{2}+\epsilon\beta} + n^{1-\left(\frac{1}{2}-\epsilon\right)(\beta-2)}\right) = \mathcal{O}\left(n^{\frac{4-\beta}{2}+\epsilon\beta}\right).
\end{aligned}$$

$\square$

By this claim, $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v \geq n^{\frac{1}{2}-2\epsilon}} \rho_v$ is smaller than $n^{\frac{1}{2}+6\epsilon}$ if λ has finite second moment, and it is smaller than $n^{\frac{4-\beta}{2}+6\epsilon}$ if λ is power law with $2 < \beta < 3$. To conclude the proof, we only have to bound $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v < n^{\frac{1}{2}-2\epsilon}} \rho_v$.

Claim: with high probability, $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v < n^{\frac{1}{2}-2\epsilon}} \rho_v < n^{\frac{1}{2}+\epsilon}$ if λ has finite second moment, $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v < n^{\frac{1}{2}-2\epsilon}} \rho_v < n^{\frac{4-\beta}{2}+\epsilon}$ if λ is power law with $2 < \beta < 3$.

Proof of claim, CM. As in the proof of Lemma 17, we can safely assume that we know the structure S of $\boldsymbol{N}^l(s)$. Let us sort the stubs in $\boldsymbol{R}^l(s)$, not paired by S, obtaining $a_1, \ldots, a_k$, and let $\boldsymbol{a}_i$ be the stub paired with a_i. Let $\operatorname{res}(a)$ be the number of stubs of the node a, minus a, and let $\boldsymbol{X}_i = \operatorname{res}(\boldsymbol{a}_i)$ if $\operatorname{res}(\boldsymbol{a}_i) < n^{\frac{1}{2}-2\epsilon}$, 0 otherwise: clearly, $\sum_{v \in \mathbf{\Gamma}^{l+1}(s), \rho_v \leq n^{\frac{1}{2}-2\epsilon}} \rho_v \leq \sum_{i=1}^k \boldsymbol{X}_i$ (with equality if there are no horizontal or diagonal edges in the BFS tree). After the first $i-1$ stubs are paired, since $i < n^{\frac{1}{2}+\epsilon}$ and since the number of stubs paired in S is $\mathcal{O}\left(n^{\frac{1}{2}+\epsilon}\log n\right)$, for each $k < n^{\frac{1}{2}-2\epsilon}$,

$$\begin{aligned}
\Pr(\boldsymbol{X}_i = k) &= \Pr(\operatorname{res}(\boldsymbol{a}_i) = k) \\
&= \frac{|\{a \in A : a \text{ unpaired after } i \text{ rounds}, \operatorname{res}(a) = k\}|}{|\{a \in A : a \text{ unpaired after } i \text{ rounds}\}|} \\
&= \frac{|\{a \in A : \operatorname{res}(a) = k\}| + \mathcal{O}\left(n^{\frac{1}{2}+\epsilon}\right)}{|A| + \mathcal{O}\left(n^{\frac{1}{2}+\epsilon}\right)} \\
&= \frac{(k+1)\lambda(k+1)}{M_1(\lambda)} + \mathcal{O}\left(n^{-\frac{1}{2}+\epsilon}\right).
\end{aligned}$$

Consequently, conditioned on all pairings of a_j for $j < i$, $\mathbb{E}[\boldsymbol{X}_i] = \sum_{k=0}^{n^{\frac{1}{2}-2\epsilon}} k\frac{(k+1)\lambda(k+1)}{M_1(\lambda)} + \mathcal{O}(n^{-\frac{1}{2}+\epsilon}\log n) = \alpha(n)$, where $\alpha(n) = \mathcal{O}(1)$ if λ has finite second moment, and $\alpha(n) =$

$\mathcal{O}(n^{\frac{3-\beta}{2}})$ if λ is power law with $2 < \beta < 3$. Hence, for each ϵ, $\sum_{i=1}^{k} \boldsymbol{X}_i - i(M_1(\lambda) + \epsilon)$ is a supermartingale, and by Azuma's inequality

$$\Pr\left(\sum_{i=1}^{k} \boldsymbol{X}_i - k\alpha(n) \geq \alpha(n)\right) \leq \exp\left(-\frac{\alpha(n)^2}{2\sum_{i=1}^{k} n^{\frac{1}{2}-2\epsilon}}\right) \leq \exp(-n^{\epsilon}).$$

Then, w.h.p., $\sum_{i=1}^{k} \boldsymbol{X}_i \leq n^{\frac{1}{2}+\epsilon}(\alpha(n)+2)$, concluding the proof of the claim. $\square$

Proof of claim, IRG. The number of nodes w with weight at most $n^{\frac{1}{2}-2\epsilon}$ that belong to $\boldsymbol{\Gamma}^{l+1}(s)$ is at most $\sum_{v \in \boldsymbol{\Gamma}^l(s), \rho_v < n^{\frac{1}{2}-2\epsilon}} \sum_{w \in V} \rho_w \boldsymbol{X}_{v,w}$, where $\boldsymbol{X}_{v,w} = 1$ with probability $f\left(\frac{\rho_v \rho_w}{M}\right) = \mathcal{O}\left(\frac{\rho_v \rho_w}{M}\right)$ because $\rho_v \rho_w < n^{1-\epsilon}$. Moreover,

$$\mathbb{E}\left[\sum_{v \in \boldsymbol{\Gamma}^l(s), \rho_v < n^{\frac{1}{2}-2\epsilon}} \sum_{w \in V} \rho_w \boldsymbol{X}_{v,w}\right] = \mathcal{O}\left(\boldsymbol{r}^l(s)\frac{\sum_{v \in V} \rho_v^2}{n}\right) = \boldsymbol{r}^l(s)\alpha(n)$$

where $\alpha(n) = \mathcal{O}(1)$ if λ has finite second moment, and $\alpha(n) = \mathcal{O}\left(n^{\frac{3-\beta}{2}}\right)$ if λ is power law with $2 < \beta < 3$.

By Hoeffding inequality,

$$\Pr\left(\sum_{v \in \boldsymbol{\Gamma}^l(s), \rho_v < n^{\frac{1}{2}-2\epsilon}} \sum_{w \in V} \rho_w \boldsymbol{X}_{v,w} - \boldsymbol{r}^l(s)\alpha(n) \geq \boldsymbol{r}^l(s)\alpha(n)\right) \leq n^{\frac{\boldsymbol{r}^l(s)\alpha(n)}{\boldsymbol{r}^l(s) n^{\frac{1}{2}-2\epsilon}}} \leq n^{-\epsilon}.$$

This concludes the proof. $\square$

This claim lets us conclude the proof of the lemma. $\square$

Proof of Theorem 4. Let $D_s^i = \sum_{v \in \boldsymbol{\Gamma}^i(s)} \deg(v)$, $D_t^j = \sum_{w \in \boldsymbol{\Gamma}^j(t)} \deg(w)$, and let us suppose that we have visited until level l_s from s, until level l_t from t, and that $D_s^{l_s}, D_t^{l_t} > n^{\frac{1}{2}+2\epsilon}$. If this situation never occurs, by Theorem 18, the total number of visited edges is at most $\mathcal{O}(\log n)n^{\frac{1}{2}+2\epsilon} = \mathcal{O}(n^{\frac{1}{2}+3\epsilon})$, and the conclusion follows. Otherwise, again by Theorem 18, the number of edges visited in the two BFS trees before levels l_s and l_t is $\mathcal{O}(n^{\frac{1}{2}+3\epsilon})$. Furthermore, by Lemma 15, $\boldsymbol{r}^{l_s}(s), \boldsymbol{r}^{l_t}(t) > n^{\frac{1}{2}+2\epsilon}$. We claim that, without loss of generality, we can assume $\boldsymbol{r}^{l_s-1}(s) < \epsilon \boldsymbol{r}^{l_s}(s)$, to apply Lemma 17. Indeed, if $\boldsymbol{r}^{l_s-1}(s)$ is too big, we iteratively decrease l_s until we find a neighbor verifying $\boldsymbol{r}^{l_s}(s) > (1-\epsilon')\boldsymbol{r}^{l_s-1}(s)$. This process can last at most $\mathcal{O}(\log n)$ steps, and hence it is stopped at a point l_s such that $\boldsymbol{r}^{l_s}(s) > n^{\frac{1}{2}+2\epsilon}(1-\epsilon')^{\mathcal{O}(\log n)} \geq n^{\frac{1}{2}+\epsilon'}$ if ϵ' is small enough. Similarly, we can suppose without loss of generality that $\boldsymbol{r}^{l_t}(t) > (1-\epsilon')\boldsymbol{r}^{l_t-1}(t)$. By Lemma 17, $d(s,t) \leq l_s + l_t + 2$, and the number of nodes needed to conclude the BFS is at most $D_s^{l_s} + D_t^{l_t}$ (note that, if we extend twice the visit from s, it means that $D_s^{l_s+1} < D_t^{l_t}$). By Lemma 15, $D_s^{l_s} \leq n^{\epsilon}\boldsymbol{r}^{l_s}(s)$, and by Lemma 19 this value is at most $n^{\frac{1}{2}+3\epsilon}$ if λ has finite second moment, and $n^{\frac{4-\beta}{2}+3\epsilon}$ if λ is power law with $2 < \beta < 3$. We conclude that the total number of visited nodes is at most $n^{\frac{1}{2}+3\epsilon} + D_s^{l_s} + D_t^{l_t} \leq n^{\frac{1}{2}+3\epsilon} + \boldsymbol{r}^{l_s}(s) + \boldsymbol{r}^{l_t}(t) \leq n^{\frac{1}{2}+4\epsilon}$ (resp., $n^{\frac{4-\beta}{2}+4\epsilon}$) if λ has finite second moment (resp., if λ is power law with $2 < \beta < 3$). The theorem follows by changing the value of ϵ. $\square$

F Detailed Experimental Results

Table 1: Detailed experimental results (undirected graphs). Empty values correspond to graphs on which the algorithm needed more than 1 hour.

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.005$									
advogato	64427	126052	174728	185998	0.193	11.450	9.557	10.498	261.2
as20000102	115797	126052	18329844	4126626	0.231	6.990	611.584	136.764	377.6
ca-GrQc	61611	146052	142982	129165	0.126	5.574	3.500	2.839	353.4
ca-HepTh	31735	146052	121587	129165	0.222	14.921	7.389	8.168	9.9
C_elegans	69729	146052	204634	185998	0.132	6.876	5.693	5.261	270.7
com-amazon.all	40711	166052	69708	74747	0.340	122.020	12.011	11.849	21.9
dip20090126_MAX	156552	166052			1.374	34.595			15354.9
D_melanogaster	51227	126052	144680	154998	0.123	19.253	15.061	16.882	520.8
email-Enron	74745	146052	257989	267838	0.280	79.296	101.529	106.278	1408.0
HC-BIOGRID	78804	146052	245780	223198	0.177	7.751	7.534	6.951	713.2
Homo_sapiens	60060	146052	156973	154998	0.151	32.078	23.716	24.449	643.8
hprd_pp	59125	146052	151499	154998	0.127	18.323	13.425	13.458	456.4
Mus_musculus	92081	146052	504669	385688	0.168	4.058	7.723	6.083	226.6
oregon1_010526	114829	126052	6798931	2865712	0.228	13.281	442.370	185.711	681.6
oregon2_010526	115764	126052	5714183	2865712	0.236	15.823	452.554	229.234	822.2
$\lambda = 0.010$									
advogato	19811	31513	47076	48243	0.081	2.804	2.576	2.788	258.2
as20000102	29062	31513	2688614	1070372	0.071	1.777	88.886	35.049	377.3
ca-GrQc	18535	36513	37529	33501	0.049	1.417	0.987	0.753	350.6
ca-HepTh	13761	36513	31721	33501	0.188	3.771	2.078	2.275	10.0
C_elegans	19888	36513	54327	48243	0.048	1.803	1.586	1.483	269.4
com-amazon.all	14641	41513	18007	19386	0.312	31.004	5.196	7.623	21.5
dip20090126_MAX	39314	41513			0.395	8.578			15301.7
D_melanogaster	15136	31513	37219	40202	0.063	4.983	3.891	4.715	519.9
email-Enron	21637	36513	65392	69471	0.198	19.877	24.997	27.296	1387.2
HC-BIOGRID	22924	36513	62413	57892	0.052	1.979	1.989	1.906	712.5
Homo_sapiens	20273	36513	41006	40202	0.085	7.876	6.442	6.636	642.7
hprd_pp	18403	36513	39994	40202	0.074	4.348	4.097	3.714	456.4
Mus_musculus	25146	36513	130384	100040	0.061	1.055	1.965	1.718	223.9
oregon1_010526	30514	31513	1104167	743313	0.087	3.254	70.383	47.740	683.3
oregon2_010526	29117	31513	954515	743313	0.088	3.983	73.942	59.103	822.1
$\lambda = 0.015$									
advogato	9570	14006	21027	22204	0.050	1.428	1.227	1.299	261.0
as20000102	13035	14006	705483	492651	0.047	0.776	22.939	16.136	377.6
ca-GrQc	8668	16228	17419	15419	0.031	0.637	0.493	0.361	345.8
ca-HepTh	7524	16228	15002	15419	0.167	1.641	0.939	1.050	11.5
C_elegans	10956	16228	25233	22204	0.034	0.782	0.740	0.732	267.6
com-amazon.all	8228	18451		15419	0.301	13.814		7.785	21.9
dip20090126_MAX	17578	18451			0.203	3.851			15197.2
D_melanogaster	9350	14006	17229	18503	0.053	2.216	1.904	2.182	519.3
email-Enron	11209	16228	29134	31974	0.170	8.845	10.510	12.423	1367.4
HC-BIOGRID	12694	16228	28805	26645	0.043	0.858	0.946	0.947	708.6
Homo_sapiens	10142	16228	18491	18503	0.072	3.717	3.076	3.061	640.4
hprd_pp	10659	16228	17969	18503	0.056	1.919	1.719	1.752	451.5
Mus_musculus	11825	16228	59756	46043	0.033	0.458	0.906	0.812	222.8
oregon1_010526	13662	14006	426845	342118	0.056	1.522	26.420	21.871	681.4
oregon2_010526	13024	14006	333638	342118	0.060	1.773	26.070	27.298	833.6

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.020$									
advogato	5874	7879	11993	12915	0.054	0.710	0.665	0.765	260.3
as20000102	7436	7879	312581	238814	0.037	0.441	10.066	7.819	376.2
ca-GrQc	5313	9129	9939	10762	0.032	0.356	0.293	0.268	347.9
ca-HepTh	5115	9129	8708	8968	0.191	0.891	0.694	0.611	10.5
C_elegans	7172	9129	14871	12915	0.030	0.439	0.436	0.439	263.5
com-amazon.all	5467	10379	12232	10762	0.331	7.683	4.338	5.459	17.9
dip20090126_MAX	9966	10379			0.148	2.165			15188.3
D_melanogaster	5610	7879	10201	10762	0.056	1.236	1.265	1.306	520.9
email-Enron	7458	9129	16443	15498	0.174	4.916	6.102	6.034	1371.7
HC-BIOGRID	8459	9129	17406	15498	0.026	0.505	0.602	0.582	716.6
Homo_sapiens	6292	9129	10481	10762	0.064	1.944	1.672	1.814	644.8
hprd_pp	6611	9129	10501	10762	0.050	1.089	0.930	1.050	449.8
Mus_musculus	7227	9129	31634	26782	0.026	0.255	0.507	0.532	221.0
oregon1_010526	7733	7879	220948	199011	0.051	0.863	13.584	12.989	679.2
oregon2_010526	7381	7879	152242	165842	0.059	1.031	11.676	13.290	836.0
$\lambda = 0.025$									
advogato	3883	5043	7439	7110	0.052	0.450	0.421	0.468	263.4
as20000102	4829	5043	130506	157779	0.033	0.285	4.097	5.108	373.5
ca-GrQc	3982	5843	6427	5925	0.028	0.242	0.180	0.162	342.1
ca-HepTh	3773	5843	6016	5925	0.176	0.573	0.374	0.416	11.8
C_elegans	4477	5843	9557	8532	0.025	0.292	0.293	0.293	266.6
com-amazon.all	4059	6643	58995	14745	0.338	4.744	9.644	7.217	21.3
dip20090126_MAX	6457	6643			0.125	1.397			15193.8
D_melanogaster	3993	5043	6279	7110	0.056	0.793	0.827	0.870	522.6
email-Enron	4576	5843	11001	12287	0.574	3.289	3.888	4.705	1381.5
HC-BIOGRID	5940	5843	11109	10239	0.029	0.321	0.414	0.404	714.0
Homo_sapiens	4796	5843	7109	7110	0.077	1.245	1.154	1.215	647.2
hprd_pp	5071	5843	6772	7110	0.052	0.687	0.579	0.647	446.3
Mus_musculus	4477	5843	18626	17694	0.026	0.168	0.302	0.385	219.8
oregon1_010526	5027	5043	92520	109568	0.058	0.516	5.762	7.014	681.0
oregon2_010526	4763	5043	86287	91306	0.050	0.638	7.140	7.420	847.5
$\lambda = 0.030$									
advogato	3256	3502	5521	5090	0.048	0.361	0.335	0.322	260.6
as20000102	3388	3502	122988	94140	0.029	0.199	3.899	3.182	378.7
ca-GrQc	2981	4057	4686	4241	0.025	0.169	0.145	0.175	344.7
ca-HepTh	2992	4057	4022	4241	0.190	0.435	0.286	0.341	7.9
C_elegans	3707	4057	6905	6108	0.026	0.198	0.218	0.217	265.9
com-amazon.all	3157	4613	39917	12668	0.330	3.631	8.491	6.852	17.5
dip20090126_MAX	4499	4613	12373086		0.300	0.972	1958.083		15199.0
D_melanogaster	2893	3502	4883	5090	0.052	0.562	0.620	0.807	510.4
email-Enron	3619	4057	7321	7330	0.172	2.735	2.724	2.806	1399.7
HC-BIOGRID	3883	4057	7499	7330	0.024	0.367	0.316	0.307	720.8
Homo_sapiens	3322	4057	4982	5090	0.066	0.897	0.842	0.877	654.2
hprd_pp	3355	4057	5028	5090	0.048	0.478	0.458	0.503	448.8
Mus_musculus	3806	4057	14290	10556	0.033	0.127	0.237	0.233	221.4
oregon1_010526	3542	3502	85854	78450	0.052	0.366	5.402	5.039	675.7
oregon2_010526	3355	3502	61841	65375	0.048	0.509	4.972	5.302	822.8

Table 2: Detailed experimental results (directed graphs). Empty values correspond to graphs on which the algorithm needed more than 1 hour.

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.005$									
as-caida20071105	103488	146052	546951	462826	0.253	35.652	96.312	85.201	1066.4
cfinder-google	137313	146052			0.820	14.190			554.4
cit-HepTh	98054	166052	481476	462826	0.579	22.651	38.339	37.720	5773.1
ego-gplus	37862	66052		2388093	0.136	6.266		11.912	1.9
ego-twitter	37125	66052		154998	0.178	6.181		4.804	2.3
freeassoc	41602	166052	89424	89697	0.116	9.384	1.036	0.997	223.5
lasagne-spanishbook	112266	146052	8918751	4126626	0.250	17.374	687.815	318.784	552.8
opsahl-openflights	73744	146052	200164	185998	0.179	6.191	5.165	4.849	431.1
p2p-Gnutella31	39193	166052	81335	89697	0.254	50.542	10.213	10.662	162.1
polblogs	71423	126052	387278	321406	0.174	1.165	3.522	3.017	190.3
soc-Epinions1	58223	146052	109607	107637	0.671	100.516	62.524	62.167	671.9
subelj-cora-cora	68112	186052	180740	185998	0.185	19.012	8.464	8.873	440.4
subelj-jdk-jdk	42361	146052	84549	89697	0.110	2.955	0.230	0.257	51.5
subelj-jung-j-jung-j	43637	126052	84225	89697	0.216	2.397	0.238	0.211	45.9
wiki-Vote	47003	126052	100153	107637	0.131	5.916	2.990	3.219	162.4
$\lambda = 0.010$									
as-caida20071105	30382	36513	132997	120048	0.135	8.902	22.251	20.315	1066.1
cfinder-google	34452	36513			0.156	3.664			553.2
cit-HepTh	27203	41513	117633	120048	0.255	5.654	8.803	9.677	5798.8
ego-gplus	13123	16513		4602412	0.085	1.584		22.510	2.3
ego-twitter	13310	16513		83366	0.086	1.518		3.500	2.2
freeassoc	13222	41513	23586	23264	0.080	2.335	0.238	0.227	220.7
lasagne-spanishbook	32527	36513	1366576	1070372	0.101	4.339	104.916	83.610	553.4
opsahl-openflights	22473	36513	52196	48243	0.059	1.475	1.348	1.339	432.0
p2p-Gnutella31	13101	41513	21567	23264	0.192	12.950	2.677	2.831	162.1
polblogs	22286	31513	101466	83366	0.046	0.298	1.078	0.834	190.6
soc-Epinions1	17061	36513	28493	27917	0.320	27.194	16.516	15.974	659.5
subelj-cora-cora	23078	46513	47936	48243	0.128	4.797	1.988	2.101	432.4
subelj-jdk-jdk	14047	36513	22038	23264	0.066	0.734	0.099	0.075	52.2
subelj-jung-j-jung-j	14894	36513	22266	23264	0.064	0.696	0.113	0.083	46.4
wiki-Vote	17380	31513	26352	27917	0.088	1.446	0.792	0.870	155.7
$\lambda = 0.015$									
as-caida20071105	14157	16228	55049	55252	0.477	3.963	8.518	8.914	1059.6
cfinder-google	15400	16228			0.123	1.666			558.1
cit-HepTh	13002	18451	47035	46043	0.232	2.529	3.807	3.766	5883.0
ego-gplus	7205	7340		2118317	0.080	0.710		12.808	2.2
ego-twitter	7403	7340	1958981	114573	0.082	0.704	14.021	5.304	2.3
freeassoc	7095	18451	10956	10707	0.297	1.072	0.115	0.110	222.0
lasagne-spanishbook	14542	16228	437041	410542	0.068	1.936	34.098	33.153	552.8
opsahl-openflights	11550	16228	24433	22204	0.034	0.649	0.643	0.648	433.9
p2p-Gnutella31	7227	18451	10002	10707	0.190	5.732	1.317	1.444	157.1
polblogs	10296	14006	46648	38369	0.029	0.136	0.516	0.435	189.5
soc-Epinions1	9273	16228	13571	12849	0.450	12.115	7.661	7.629	662.0
subelj-cora-cora	11297	20673	20940	22204	0.502	2.135	0.937	1.073	445.6
subelj-jdk-jdk	8360	14006	10045	10707	0.052	0.288	0.080	0.049	51.6
subelj-jung-j-jung-j	8712	16228	10319	10707	0.046	0.312	0.068	0.042	45.6
wiki-Vote	8668	14006	12406	12849	0.408	0.659	0.380	0.429	152.6

	Number of iterations				Time (s)				Edges
Graph	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA	RK	ABRA-Aut	ABRA-1.2	KADABRA
$\lambda = 0.020$									
as-caida20071105	9086	9129	31242	32139	0.104	2.226	4.954	5.087	1064.2
cfinder-google	8745	9129			0.353	0.946			551.9
cit-HepTh	8679	10379	27755	32139	1.249	1.442	2.225	2.684	5758.0
ego-gplus	4785	4129		1478684	0.081	0.395		9.234	2.6
ego-twitter	4950	7879		138201	0.083	0.743		5.079	2.4
freeassoc	4268	10379	6509	6227	0.065	0.609	0.078	0.073	216.4
lasagne-spanishbook	8338	9129	294793	286577	0.058	1.074	22.405	22.468	555.0
opsahl-openflights	7392	9129	14202	12915	0.029	0.364	0.390	0.391	432.3
p2p-Gnutella31	4697	10379	5700	6227	0.190	3.162	0.695	0.816	156.7
polblogs	6325	7879	25593	22318	0.023	0.076	0.283	0.252	188.4
soc-Epinions1	5489	9129	7686	7473	0.457	6.738	4.506	4.335	651.8
subelj-cora-cora	6325	11629	12437	12915	0.500	1.203	0.571	0.520	450.8
subelj-jdk-jdk	5456	9129	6070	6227	0.191	0.192	0.062	0.044	52.3
subelj-jung-j-jung-j	5643	9129		6227	0.217	0.176		0.045	46.6
wiki-Vote	4939	7879	7125	7473	0.075	0.368	0.221	0.259	152.2
$\lambda = 0.025$									
as-caida20071105	5723	5843	21020	21233	0.022	1.465	3.129	3.340	1093.4
cfinder-google	6275	5843			0.019	0.648			758.0
cit-HepTh	5206	6643	15915	21233	0.034	0.940	1.351	1.891	6130.5
ego-gplus	2989	5043		4200646	0.013	0.485		20.309	2.6
ego-twitter	2958	2643		157779	0.012	0.248		6.291	2.4
freeassoc	2804	6643	4285	4114	0.009	0.399	0.061	0.058	261.5
lasagne-spanishbook	5409	5043	129999	131482	0.013	0.592	10.040	10.221	626.1
opsahl-openflights	4557	5843	10116	8532	0.009	0.236	0.290	0.267	561.3
p2p-Gnutella31	3069	6643	3931	4114	0.043	2.149	0.590	0.663	176.8
polblogs	3880	5043	15986	14745	0.007	0.049	0.185	0.176	241.9
soc-Epinions1	3689	5843	5060	4937	0.188	4.158	2.798	2.791	888.1
subelj-cora-cora	5264	7443	7699	8532	0.020	0.781	0.360	0.408	436.5
subelj-jdk-jdk	3201	5843	9428	4937	0.008	0.122	0.065	0.036	57.2
subelj-jung-j-jung-j	3168	5043	13471	5925	0.007	0.098	0.057	0.045	57.7
wiki-Vote	3265	5043	4566	4937	0.009	0.241	0.137	0.178	174.7
$\lambda = 0.030$									
as-caida20071105	3956	4057	12696	15202	0.017	1.029	1.973	2.434	1285.2
cfinder-google	4419	4057			0.013	0.412			770.5
cit-HepTh	4062	4613	13172	12668	0.033	0.672	1.195	1.059	6131.6
ego-gplus	2434	1835		4330990	0.009	0.188		21.395	3.1
ego-twitter	2270	1835	98839	135562	0.008	0.174	4.909	5.510	2.2
freeassoc	2105	4613	3008	3534	0.006	0.285	0.101	0.091	250.7
lasagne-spanishbook	3820	4057	158028	94140	0.010	0.487	12.564	7.855	656.8
opsahl-openflights	3450	4057	6556	6108	0.007	0.165	0.184	0.195	481.4
p2p-Gnutella31	2367	4613	2874	2945	0.036	1.412	0.422	0.445	166.5
polblogs	3567	3502	11357	8796	0.007	0.036	0.151	0.122	207.9
soc-Epinions1	2659	4057	3585	3534	0.312	3.211	2.186	2.046	918.3
subelj-cora-cora	3790	5169	5681	5090	0.016	0.564	0.272	0.265	422.6
subelj-jdk-jdk	2425	4057	25575	5090	0.006	0.097	0.100	0.064	57.4
subelj-jung-j-jung-j	2436	3502	43584	5090	0.006	0.079	0.140	0.059	57.0
wiki-Vote	2633	3502	3467	3534	0.006	0.188	0.148	0.149	188.2

G Wikipedia and IMDB Results

In this section, we report our results on the Wikipedia citation network, and on all snapshots of the IMDB actors collaboration network. In the ranking column, we report one number if the position in the ranking is guaranteed with probability 0.9, otherwise we report a lower and an upper bound, which hold with the same probability.

We remark that, as for the IMDB database, the top-k betweenness centralities of a single snapshot of a similar graph (`hollywood-2009` in [10]) have been previously computed exactly, with one week of computation on a 40-core machine [47].

G.1 The Results on the IMDB Graph

In 2014, the most central actor is Ron Jeremy, who is listed in the Guinness Book of World Records for "Most Appearances in Adult Films", with more than 2000 appearances. Among his non-adult ones, we mention *The Godfather Part III*, *Ghostbusters*, *Crank: High Voltage* and *Family Guy*[3]. His topmost centrality in the actor collaboration network has been previously observed by similar experiments on betweenness centrality [47]. Indeed, around 3 actors out of 100 in the IMDB database played in adult movies, which explains why the

[3]The latter is a TV-series, which are not taken into account in our data.

high number of appearances of Ron Jeremy both in the adult and non-adult film industry rises his betweenness to the top.

The second most-central actor is Lloyd Kaufman, which is best known as a co-founder of *Troma Entertainment Film Studio* and as the director of many of their feature films, including the cult movie *The Toxic Avenger*. His high betweenness score is likely due to his central role in the low-budget independent film industry.

The third "actor" is the historical German dictator Adolf Hitler, since his appearances in several historical footages, that were re-used in several movies (e.g. in *The Imitation Game*), are credited by IMDB as cameo role. Indeed, he appears among the topmost actors since the 1984 snapshot, being the first one in the 1989 and 1994 ones, and during those years many movies about the World War II were produced.

Observe that the betweenness centrality measure on our graph does not discriminate between important and marginal roles. For example, the actress Bess Flowers, who appears among the top actors in the snapshots from 1959 to 1979, rarely played major roles, but she appeared in over 700 movies in her 41 years career.

G.2 The Results on the Wikipedia Graph

All topmost pages in the betweenness centrality ranking, except for the World War II, are countries. This is not surprising if we consider that, for most topics (such as important people or events), the corresponding Wikipedia page refers to their geographical context (since it mentions the country of origin of the given person or where a given event took place). It is also worth noting the correlation between the high centrality of the *World War II* Wikipedia page and that of Adolf Hitler in the IMDB graph.

Interestingly, a similar ranking is obtained by considering the closeness centrality measure in the inverse graph, where a link from page p_1 to page p_2 exists if a link to page p_1 appears in page p_2 [8]. However, in contrast with the results in [8] when edges are oriented in the usual way, the pages about specific years do not appear in the top ranking. We note that the betweenness centrality of a node in a directed graph does not change if the orientation of all edges is flipped.

Finally, the most important pages is the United States, confirming a common conjecture. Indeed, in `http://wikirank.di.unimi.it/`, it is shown that the United States are the center according to harmonic centrality, and many other measures. Further evidence for this conjecture comes from the Six Degree of Wikipedia game (`http://thewikigame.com/6-degrees-of-wikipedia`), where a player is asked to go from one page to the other following the smallest possible number of links: a hard variant of this game forces the player not to pass from the *United States* page, which is considered to be central. Our results thus confirm that the conjecture is indeed true for the betweenness centrality measure.

Table 3: The top-k betweenness centralities of the Wikipedia graph computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Wikipedia page	Lower bound	Estimated betweenness	Upper bound
1)	United States	0.046278	0.047173	0.048084
2)	France	0.019522	0.020103	0.020701
3)	United Kingdom	0.017983	0.018540	0.019115
4)	England	0.016348	0.016879	0.017428
5-6)	Poland	0.012092	0.012287	0.012486
5-6)	Germany	0.011930	0.012124	0.012321
7)	India	0.009683	0.010092	0.010518
8-12)	World War II	0.008870	0.009065	0.009265
8-12)	Russia	0.008660	0.008854	0.009053
8-12)	Italy	0.008650	0.008845	0.009045
8-12)	Canada	0.008624	0.008819	0.009018
8-12)	Australia	0.008620	0.008814	0.009013

Table 4: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1939 (69011 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Meyer, Torben	0.022331	0.022702	0.023049
2)	Roulien, Raul	0.021361	0.021703	0.022071
3)	Myzet, Rudolf	0.014229	0.014525	0.014747
4)	Sten, Anna	0.013245	0.013460	0.013723
5)	Negri, Pola	0.012509	0.012768	0.012943
6-7)	Jung, Shia	0.012250	0.012379	0.012509
6-7)	Ho, Tai-Hau	0.012195	0.012324	0.012454
8)	Goetzke, Bernhard	0.010721	0.010978	0.011201
9-10)	Yamamoto, Togo	0.010095	0.010224	0.010354
9-10)	Kamiyama, Sōjin	0.010087	0.010215	0.010344

Table 5: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1944 (83068 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Meyer, Torben	0.018320	0.018724	0.019136
2)	Kamiyama, Sōjin	0.012629	0.012964	0.013308
3-4)	Jung, Shia	0.010751	0.010901	0.011053
3-4)	Ho, Tai-Hau	0.010704	0.010854	0.011005
5)	Myzet, Rudolf	0.010365	0.010514	0.010666
6-7)	Sten, Anna	0.009778	0.009928	0.010080
6-7)	Goetzke, Bernhard	0.009766	0.009915	0.010066
8)	Yamamoto, Togo	0.009108	0.009327	0.009539
9)	Parìs, Manuel	0.008649	0.008859	0.009108
10)	Hayakawa, Sessue	0.007916	0.008158	0.008369

Table 6: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1949 (97824 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Meyer, Torben	0.016139	0.016679	0.017236
2)	Kamiyama, Sōjin	0.012351	0.012822	0.013312
3)	Parìs, Manuel	0.011104	0.011552	0.011861
4)	Yamamoto, Togo	0.010342	0.010639	0.011086
5-6)	Jung, Shia	0.008926	0.009120	0.009318
5-6)	Goetzke, Bernhard	0.008567	0.008762	0.008962
7-9)	Paananen, Tuulikki	0.008147	0.008341	0.008539
7-9)	Sten, Anna	0.007969	0.008164	0.008363
7-9)	Mayer, Ruby	0.007967	0.008162	0.008362
10-12)	Ho, Tai-Hau	0.007538	0.007732	0.007930
10-12)	Hayakawa, Sessue	0.007399	0.007593	0.007792
10-12)	Haas, Hugo (I)	0.007158	0.007352	0.007552

Table 7: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1954 (120430 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Meyer, Torben	0.013418	0.013868	0.014334
2)	Kamiyama, Sōjin	0.010331	0.010726	0.011089
3-4)	Ertugrul, Muhsin	0.009956	0.010141	0.010331
3-4)	Jung, Shia	0.009643	0.009826	0.010013
5-6)	Singh, Ram (I)	0.008657	0.008841	0.009030
5-6)	Paananen, Tuulikki	0.008383	0.008567	0.008755
7-9)	Parìs, Manuel	0.007886	0.008070	0.008257
7-10)	Goetzke, Bernhard	0.007802	0.007987	0.008176
7-10)	Yamaguchi, Shirley	0.007531	0.007716	0.007905
8-10)	Hayakawa, Sessue	0.007473	0.007657	0.007845

Table 8: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1959 (146253 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1-2)	Singh, Ram (I)	0.010683	0.010877	0.011075
1-2)	Frees, Paul	0.010372	0.010566	0.010763
3)	Meyer, Torben	0.009478	0.009821	0.010235
4-5)	Jung, Shia	0.008623	0.008816	0.009013
4-5)	Ghosh, Sachin	0.008459	0.008651	0.008847
6-7)	Myzet, Rudolf	0.007085	0.007278	0.007476
6-7)	Yamaguchi, Shirley	0.006908	0.007101	0.007299
8)	de Còrdova, Arturo	0.006391	0.006582	0.006778
9-11)	Kamiyama, Sōjin	0.005861	0.006054	0.006254
9-12)	Paananen, Tuulikki	0.005810	0.006003	0.006202
9-12)	Flowers, Bess	0.005620	0.005813	0.006012
10-12)	Parìs, Manuel	0.005442	0.005635	0.005835

Table 9: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1964 (174826 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Frees, Paul	0.013140	0.013596	0.014067
2)	Meyer, Torben	0.007279	0.007617	0.007856
3-4)	Harris, Sam (II)	0.006813	0.006967	0.007124
3-5)	Myzet, Rudolf	0.006696	0.006849	0.007005
4-5)	Flowers, Bess	0.006422	0.006572	0.006726
6)	Kong, King (I)	0.005909	0.006104	0.006422
7)	Yuen, Siu Tin	0.005114	0.005264	0.005420
8)	Miller, Marvin (I)	0.004708	0.004859	0.005015
9-12)	de Còrdova, Arturo	0.004147	0.004299	0.004457
9-18)	Haas, Hugo (I)	0.003888	0.004039	0.004197
9-18)	Singh, Ram (I)	0.003854	0.004004	0.004160
9-18)	Kamiyama, Sōjin	0.003848	0.003999	0.004155
10-18)	Sauli, Anneli	0.003827	0.003978	0.004135
10-18)	King, Walter Woolf	0.003774	0.003923	0.004078
10-18)	Vanel, Charles	0.003716	0.003867	0.004024
10-18)	Kowall, Mitchell	0.003684	0.003834	0.003990
10-18)	Holmes, Stuart	0.003603	0.003752	0.003907
10-18)	Sten, Anna	0.003582	0.003733	0.003890

Table 10: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1969 (210527 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Frees, Paul	0.010913	0.011446	0.012005
2-3)	Yuen, Siu Tin	0.006157	0.006349	0.006547
2-3)	Tamiroff, Akim	0.006097	0.006291	0.006490
4-6)	Meyer, Torben	0.005675	0.005869	0.006069
4-7)	Harris, Sam (II)	0.005639	0.005830	0.006027
4-8)	Rubener, Sujata	0.005427	0.005618	0.005815
5-8)	Myzet, Rudolf	0.005253	0.005444	0.005641
6-8)	Flowers, Bess	0.005136	0.005328	0.005526
9-10)	Kong, King (I)	0.004354	0.004544	0.004741
9-10)	Sullivan, Elliott	0.004208	0.004398	0.004596

Table 11: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1974 (257896 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Frees, Paul	0.008507	0.008958	0.009295
2)	Chen, Sing	0.007734	0.008056	0.008507
3)	Welles, Orson	0.006115	0.006497	0.006903
4-5)	Loren, Sophia	0.005056	0.005221	0.005392
4-7)	Rubener, Sujata	0.004767	0.004933	0.005106
5-8)	Harris, Sam (II)	0.004628	0.004795	0.004967
5-8)	Tamiroff, Akim	0.004625	0.004790	0.004962
6-10)	Meyer, Torben	0.004382	0.004548	0.004720
8-12)	Flowers, Bess	0.004259	0.004425	0.004598
8-12)	Yuen, Siu Tin	0.004229	0.004397	0.004571
9-12)	Carradine, John	0.004026	0.004192	0.004364
9-12)	Myzet, Rudolf	0.003984	0.004151	0.004325

Table 12: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1979 (310278 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Chen, Sing	0.007737	0.008220	0.008647
2)	Frees, Paul	0.006852	0.007255	0.007737
3-5)	Welles, Orson	0.004894	0.005075	0.005263
3-6)	Carradine, John	0.004623	0.004803	0.004989
3-6)	Loren, Sophia	0.004614	0.004796	0.004985
4-6)	Rubener, Sujata	0.004284	0.004464	0.004651
7-17)	Tamiroff, Akim	0.003516	0.003696	0.003885
7-17)	Meyer, Torben	0.003479	0.003657	0.003844
7-17)	Quinn, Anthony (I)	0.003447	0.003626	0.003815
7-17)	Flowers, Bess	0.003446	0.003625	0.003815
7-17)	Mitchell, Gordon (I)	0.003417	0.003596	0.003785
7-17)	Sullivan, Elliott	0.003371	0.003551	0.003740
7-17)	Rietty, Robert	0.003368	0.003547	0.003735
7-17)	Tanba, Tetsurō	0.003360	0.003537	0.003724
7-17)	Harris, Sam (II)	0.003331	0.003510	0.003699
7-17)	Lewgoy, Josè	0.003223	0.003402	0.003590
7-17)	Dalio, Marcel	0.003185	0.003364	0.003553

Table 13: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1984 (375322 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Chen, Sing	0.007245	0.007716	0.008218
2-4)	Welles, Orson	0.005202	0.005391	0.005587
2-4)	Frees, Paul	0.005174	0.005363	0.005559
2-5)	Hitler, Adolf	0.004906	0.005094	0.005290
4-6)	Carradine, John	0.004744	0.004932	0.005127
5-7)	Mitchell, Gordon (I)	0.004418	0.004606	0.004802
6-8)	Jürgens, Curd	0.004169	0.004356	0.004551
7-8)	Kinski, Klaus	0.003938	0.004123	0.004318
9-12)	Rubener, Sujata	0.003396	0.003585	0.003785
9-12)	Lee, Christopher (I)	0.003391	0.003576	0.003771
9-12)	Loren, Sophia	0.003357	0.003542	0.003738
9-12)	Harrison, Richard (II)	0.003230	0.003417	0.003614

Table 14: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1989 (463078 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1-2)	Hitler, Adolf	0.005282	0.005467	0.005658
1-3)	Chen, Sing	0.005008	0.005192	0.005382
2-4)	Carradine, John	0.004648	0.004834	0.005027
3-4)	Harrison, Richard (II)	0.004515	0.004697	0.004887
5-6)	Welles, Orson	0.004088	0.004271	0.004462
5-9)	Mitchell, Gordon (I)	0.003766	0.003948	0.004139
6-9)	Kinski, Klaus	0.003691	0.003874	0.004065
6-11)	Lee, Christopher (I)	0.003610	0.003793	0.003984
6-11)	Frees, Paul	0.003582	0.003766	0.003960
8-13)	Jürgens, Curd	0.003306	0.003486	0.003676
8-13)	Pleasence, Donald	0.003299	0.003479	0.003670
10-13)	Mitchell, Cameron (I)	0.003105	0.003285	0.003476
10-13)	von Sydow, Max (I)	0.002982	0.003161	0.003350

Table 15: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1994 (557373 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Hitler, Adolf	0.005227	0.005676	0.006164
2-6)	Harrison, Richard (II)	0.003978	0.004165	0.004362
2-6)	von Sydow, Max (I)	0.003884	0.004069	0.004264
2-7)	Lee, Christopher (I)	0.003718	0.003907	0.004106
2-7)	Carradine, John	0.003696	0.003883	0.004079
2-7)	Chen, Sing	0.003683	0.003871	0.004068
4-10)	Jeremy, Ron	0.003336	0.003524	0.003722
7-11)	Pleasence, Donald	0.003253	0.003439	0.003637
7-11)	Rey, Fernando (I)	0.003234	0.003420	0.003617
7-15)	Smith, William (I)	0.003012	0.003199	0.003397
8-15)	Welles, Orson	0.002885	0.003072	0.003271
10-15)	Mitchell, Gordon (I)	0.002851	0.003036	0.003232
10-15)	Kinski, Klaus	0.002705	0.002890	0.003087
10-15)	Mitchell, Cameron (I)	0.002671	0.002858	0.003058
10-15)	Quinn, Anthony (I)	0.002640	0.002826	0.003026

Table 16: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 1999 (681358 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Jeremy, Ron	0.007380	0.007913	0.008484
2)	Hitler, Adolf	0.004601	0.005021	0.005480
3-4)	Lee, Christopher (I)	0.003679	0.003849	0.004028
3-4)	von Sydow, Max (I)	0.003604	0.003775	0.003953
5-6)	Harrison, Richard (II)	0.003041	0.003211	0.003390
5-7)	Carradine, John	0.002943	0.003114	0.003296
6-11)	Chen, Sing	0.002662	0.002834	0.003018
7-14)	Rey, Fernando (I)	0.002569	0.002740	0.002922
7-14)	Smith, William (I)	0.002559	0.002729	0.002910
7-14)	Pleasence, Donald	0.002556	0.002725	0.002906
7-14)	Sutherland, Donald (I)	0.002449	0.002617	0.002796
8-14)	Quinn, Anthony (I)	0.002307	0.002476	0.002658
8-14)	Mastroianni, Marcello	0.002271	0.002440	0.002621
8-14)	Saxon, John	0.002251	0.002420	0.002602

Table 17: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 2004 (880032 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Jeremy, Ron	0.010653	0.011370	0.012136
2)	Hitler, Adolf	0.005333	0.005840	0.006396
3-4)	von Sydow, Max (I)	0.003424	0.003608	0.003802
3-4)	Lee, Christopher (I)	0.003403	0.003587	0.003781
5-6)	Kier, Udo	0.002898	0.003081	0.003275
5-8)	Keitel, Harvey (I)	0.002646	0.002828	0.003023
6-12)	Hopper, Dennis	0.002424	0.002607	0.002804
6-16)	Smith, William (I)	0.002322	0.002504	0.002700
7-17)	Sutherland, Donald (I)	0.002241	0.002422	0.002617
7-23)	Carradine, David	0.002149	0.002329	0.002526
7-23)	Carradine, John	0.002147	0.002328	0.002524
7-23)	Harrison, Richard (II)	0.002054	0.002234	0.002430
8-23)	Sharif, Omar	0.002043	0.002222	0.002418
8-23)	Steiger, Rod	0.001988	0.002165	0.002358
8-23)	Quinn, Anthony (I)	0.001974	0.002151	0.002344
8-23)	Depardieu, Gèrard	0.001966	0.002148	0.002346
9-23)	Sheen, Martin	0.001913	0.002093	0.002291
10-23)	Rey, Fernando (I)	0.001866	0.002044	0.002238
10-23)	Kane, Sharon	0.001857	0.002038	0.002237
10-23)	Pleasence, Donald	0.001859	0.002037	0.002232
10-23)	Skarsgård, Stellan	0.001848	0.002026	0.002221
10-23)	Mueller-Stahl, Armin	0.001789	0.001969	0.002166
10-23)	Hong, James (I)	0.001780	0.001957	0.002152

Table 18: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken at the end of 2009 (1237879 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Jeremy, Ron	0.010531	0.011237	0.011991
2)	Hitler, Adolf	0.005500	0.006011	0.006568
3-4)	Kaufman, Lloyd	0.003620	0.003804	0.003997
3-4)	Kier, Udo	0.003472	0.003654	0.003845
5-6)	Lee, Christopher (I)	0.003056	0.003240	0.003435
5-8)	Carradine, David	0.002866	0.003050	0.003245
6-8)	Keitel, Harvey (I)	0.002659	0.002840	0.003034
6-9)	von Sydow, Max (I)	0.002532	0.002713	0.002907
8-13)	Hopper, Dennis	0.002237	0.002419	0.002616
9-15)	Skarsgård, Stellan	0.002153	0.002333	0.002529
9-15)	Depardieu, Gèrard	0.002001	0.002181	0.002377
9-15)	Hauer, Rutger	0.001894	0.002074	0.002271
9-15)	Sutherland, Donald (I)	0.001875	0.002054	0.002250
10-15)	Smith, William (I)	0.001811	0.001990	0.002186
10-15)	Dafoe, Willem	0.001805	0.001986	0.002186

Table 19: The top-k betweenness centralities of a snapshot of the IMDB collaboration network taken in 2014 (1797446 nodes), computed by KADABRA with $\delta = 0.1$ and $\lambda = 0.0002$.

Ranking	Actor	Lower bound	Estimated betweenness	Upper bound
1)	Jeremy, Ron	0.009360	0.010058	0.010808
2)	Kaufman, Lloyd	0.005936	0.006492	0.007100
3)	Hitler, Adolf	0.004368	0.004844	0.005373
4-6)	Kier, Udo	0.003250	0.003435	0.003631
4-6)	Roberts, Eric (I)	0.003178	0.003362	0.003557
4-6)	Madsen, Michael (I)	0.003120	0.003305	0.003501
7-9)	Trejo, Danny	0.002652	0.002835	0.003030
7-9)	Lee, Christopher (I)	0.002551	0.002734	0.002931
7-12)	Estevez, Joe	0.002350	0.002534	0.002732
9-17)	Carradine, David	0.002116	0.002296	0.002492
9-17)	von Sydow, Max (I)	0.002023	0.002206	0.002405
9-17)	Keitel, Harvey (I)	0.001974	0.002154	0.002352
10-17)	Skarsgård, Stellan	0.001945	0.002125	0.002323
10-17)	Dafoe, Willem	0.001899	0.002080	0.002279
10-17)	Hauer, Rutger	0.001891	0.002071	0.002269
10-17)	Depardieu, Gèrard	0.001763	0.001943	0.002142
10-17)	Rochon, Debbie	0.001745	0.001926	0.002126